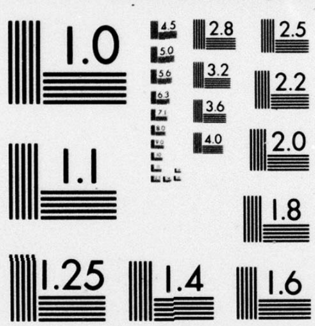
AD-A060 987 NEW MEXICO UNIV ALBUQUERQUE ERIC H WANG CIVIL ENGINE--ETC F/G 13/2
AIR FORCE REFUSE-COLLECTION SCHEDULING PROGRAM DESCRIPTION. VOL--ETC(U)
JUL 78 H J IUZZOLINO, P STANS F29601-76-C-0015

UNCLASSIFIED CERF-EE-23 CEEDO-TR-78-23-VOL-4 NL

1 OF 2
AD
A060987


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

CEEDO-TR-78-23

986-V3

2 NW

AIR FORCE REFUSE-COLLECTION SCHEDULING PROGRAM DESCRIPTION
VOLUME IV : PROGRAM PHASE 4

LEVEL III

HAROLD J. IUZZOLINO

ERIC H. WANG CIVIL ENGINEERING RESEARCH FACILITY
UNIVERSITY OF NEW MEXICO
BOX 25, UNIVERSITY STATION
ALBUQUERQUE, NEW MEXICO 87131

JULY 1978

FINAL REPORT FOR PERIOD JANUARY 1976 – APRIL 1977

Approved for public release; distribution unlimited

CEEDO

CIVIL AND ENVIRONMENTAL ENGINEERING DEVELOPMENT OFFICE
(AIR FORCE SYSTEMS COMMAND)
TYNDALL AIR FORCE BASE
FLORIDA 32403

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER 18 CEEDO-TR-78-23-Vol[redacted]-4	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER 9
4. TITLE (and Subtitle) 6 AIR FORCE REFUSE-COLLECTION SCHEDULING PROGRAM DESCRIPTION. Volume IV. Program PHASE4.		5. TYPE OF REPORT & PERIOD COVERED Final Report, Janu[redacted]y 1976 to Apr[redacted] 1977 6. PERFORMING ORG. REPORT NUMBER 14 CERF-EE-23
7. AUTHOR(s) 10 Harold J. Iuzzolino Patricia Stans		8. CONTRACT OR GRANT NUMBER(s) 15 F29601-76-C-0015
9. PERFORMING ORGANIZATION NAME AND ADDRESS Eric H. Wang Civil Engineering Research Facility, University of New Mexico, Box 25, University Station, Albuquerque, NM 87131		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS S.S. 4.03
11. CONTROLLING OFFICE NAME AND ADDRESS DET 1 (CEEDO) HQ ADTC Air Force Systems Command Tyndall Air Force Base, FL 32403		12. REPORT DATE 11 Jul[redacted]1978 13. NUMBER OF PAGES 166
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 12 161p.		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Available in DDC.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Production of maps and printed schedules
Trip pairing to minimize both total travel distance and maximum route time

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes program PHASE4, the fourth of four programs in the Air Force Refuse-Collection Scheduling Program. Program logic, input, output, and limitations are presented in detail. Some recommendations for changes, a program listing, and sample input and output are included.

78 11 06 118

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

400976

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

PREFACE

This report documents work performed during the period January 1976 through April 1977 by the University of New Mexico under Contract F29601-76-C-0015 with DET 1 (CEEDO) ADTC, Air Force Systems Command, Tyndall Air Force Base, Florida 32403. Captain Robert F. Olfenbuttel managed the program.

This volume, which documents program PHASE4, is the fourth of four volumes constituting the Air Force refuse-collection-scheduling program description. Except for Shell's sorting algorithm, used in subroutine SHLSRT, all algorithms used in program PHASE4 were developed and coded by Harold J. Iuzzolino.

The report has been reviewed by the Information Officer and is releasable to the National Technical Information Service (NTIS). At NTIS it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

ROBERT F. OLFENBUTTEL, Capt, USAF, BSC
Chief, Resources Conservation Branch

PETER A. CROWLEY, Maj, USAF, BSC
Director of Environics

EMIL C. FREIN, Maj, USAF
Chief, Envmtl Engrg & Energy Research Division

JOSEPH S. PIZZUTO, Col, USAF, BSC
Commander

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SECTION I

INTRODUCTION

1. BACKGROUND

In designing the Air Force Refuse-Collection Scheduling Program (RCSP), the fundamental objective was to reduce collection costs. The most effective way to significantly reduce cost is to reduce the number of trips used to service a given region. If a collection crew can be dropped from the fleet, the cost of manpower will be cut. In addition, fuel and maintenance costs will be lowered if the total mileage traveled by the fleet can be reduced.

2. OBJECTIVE

The primary purpose of program PHASE4 is to produce maps and printed schedules that allow for easy implementation of the routes generated by the RCSP. PHASE4 also selects trips from the two choices per section produced by PHASE3 in a manner that provides the shortest total travel distance. PHASE4 then selects two trips for each vehicle in a manner that gives the shortest maximum route time.

3. SCOPE

This report describes the workings of program PHASE4. A program overview is given, followed by a thorough description of the logic involved in schedule generation. Flow charts provide a skeleton of the logic flow. Input and output files are described. Program requirements and restrictions, error messages and error handling techniques, definitions of important variables, and an estimate of running time are also provided.

(The reverse of this page is blank.)

SECTION II
PROGRAM OVERVIEW

Program PHASE4 performs four basic tasks. It reads the input data and performs some validity checking, pairs trips to minimize both the total travel distance and the maximum route time, prints the collection schedule, and produces maps of the collection trips. When trip information that does not come from program PHASE3 (e.g., data describing routes already in use) is used as input, the two choices of trips for each section, which PHASE3 provides, will not be available. In this case, PHASE4 uses the value of a trip-pairing indicator to determine whether the trips should be scheduled as single-trip routes or should be paired sequentially into two-trip routes.

The input to PHASE4 consists of data files passed on from previous programs and data read in from cards. The file assignments are as follows: TAPE1 is the segment data, TAPE2 is the node data, TAPE3 is the street-name data, and TAPE9 is the path data. The data to be read from cards consist of a title, the unit of refuse, a trip-pairing indicator, street numbers and names, time limits for trips and the times and durations of breaks, vehicle descriptions, and map bounds. The street numbers and names are optional if they are on file TAPE3. The map-bounds record is optional and may be used to describe which regions should be plotted for a particular section and trip. If the route description is not found on TAPE9, then it must be read from cards.

The program consists of a main program named PHASE4, 11 subroutines, and 2 function subprograms. PHASE4 reads the title and refuse units from cards and the street segment and node data from files TAPE1 and TAPE2. It then calls STRIN, which looks for street names on file TAPE3. If the street names are not found on TAPE3, they are read from cards. Control returns to PHASE4, which reads the starting times and durations for lunch and two breaks and the vehicle capacities and identifications. PHASE4 then looks for map-bounds cards. If the map bounds are omitted, travel will be shown in only the collection region. PHASE4 reads the route description from file TAPE9, which has been generated by PHASE3.

If route descriptions are not found on TAPE9 or on cards, the program terminates. Subroutine CUMTD is called to determine the cumulative time and distance for travel from the garage to the collection region, within the collection region, and from the region to the landfill. CUMTD also totals the refuse accumulated for each trip. A summary of the time, distance, and refuse quantity for each section is printed by PHASE4. The trips are then ordered by SHLSRT according to vehicle capacity. Appropriate morning and afternoon trips are paired in such a way as to minimize the total distance traveled. The section numbers comprising the routes are stored by program PHASE4. TAPE9 is then rewound, and POSIT9 is called to position the tape at the correct trip data. The route data are read from TAPE9, and PRSCHED is called to print the schedule. PHASE4 then scans the map-bounds data. Each time bounds other than zero are found, PATHPLT is called to plot the appropriate vehicle path. Subroutine STNAME appends the street names before the map is drawn. MAPPLT then plots the map, and control returns to PHASE4.

The flow of control from one subprogram to another is shown in Figure 1. Within each subprogram, only the first call to each other subprogram is shown. (Subroutines PLOTS, PLOT, and SYMBOL are system subroutines and are not included in the description of PHASE4.)

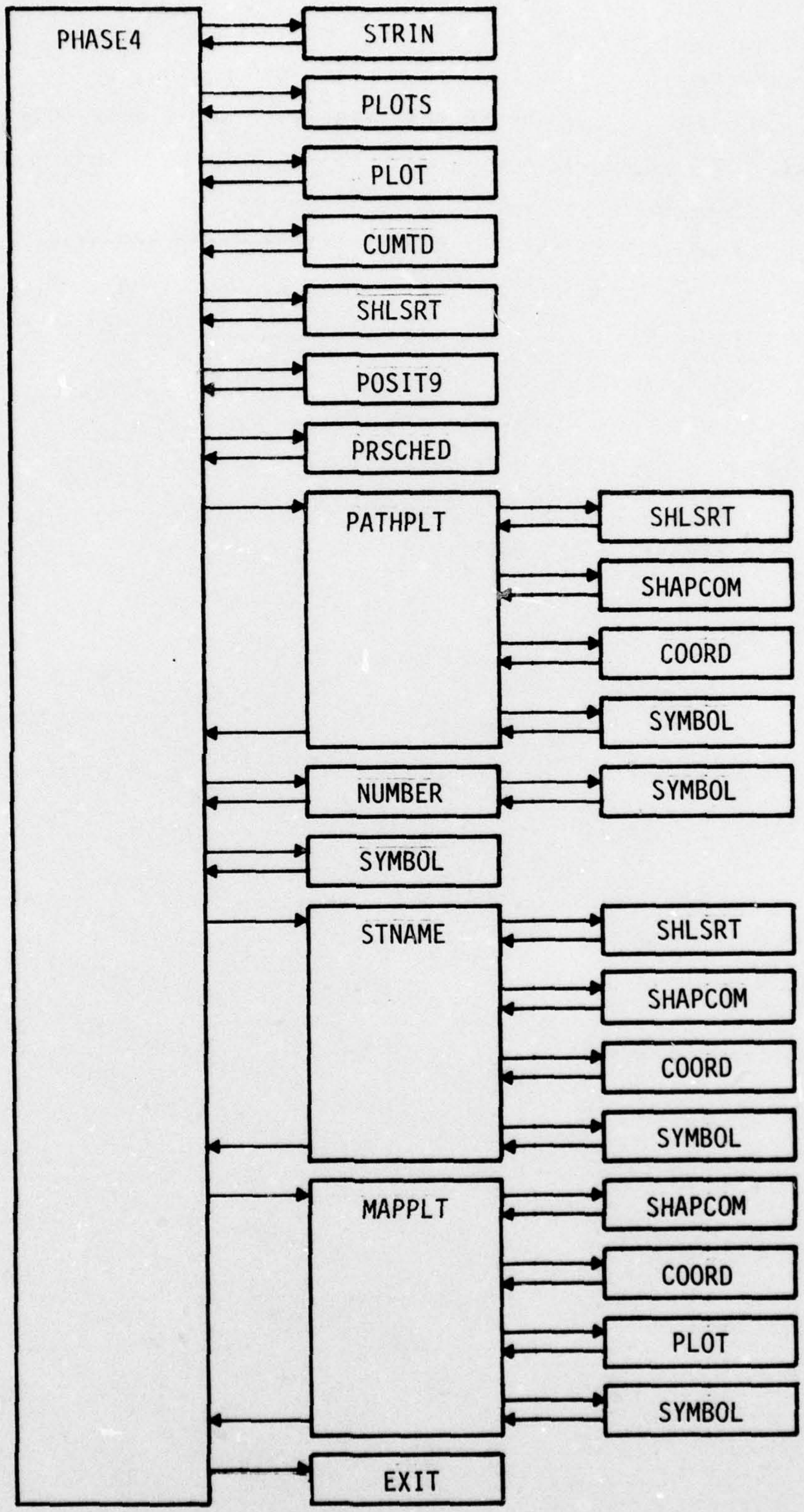


Figure 1. Control Relationships Among Subprograms

(The reverse of this page is blank.)

SECTION III
PROGRAM LOGIC

The logic for program PHASE4 is described from three viewpoints. The first description is task-oriented. The second describes the storage and use of input data. The third describes each subroutine and the main program in terms of its purpose and the manipulations performed within it.

1. PROGRAM TASKS

The processing performed by program PHASE4 can be grouped into four tasks: checking the input data, pairing the trips, generating the printed schedule, and plotting the route maps.

Trip pairing is accomplished by the statements in the main program, from the loop through statement 530 through statement 620. Vehicle capacities for the trips are sorted into increasing order; the sequence numbers of the trips are carried along during the sort. The distances saved by using the second trip of the day rather than the first trip are computed and sorted into increasing order within each vehicle capacity. The sequence numbers of the trips are carried along during the sort. The trip starting at the garage will be used for the half of the sections that provides the smaller distance savings. The trip starting at the landfill will be used for the remaining half of the sections. This procedure causes half of the sections to be serviced by trips starting at the garage and half to be serviced by trips starting at the landfill and causes the total travel distance for all sections to be the minimum obtainable from the two choices of trips for each section.

The lengths of all trips starting at the garage for a given capacity vehicle are sorted into increasing order. The lengths of the trips starting at the landfill are also sorted into increasing order. The shortest trip starting at the garage and the longest trip starting at the landfill are assigned to one vehicle. The next shortest trip starting at the garage and the next longest trip starting at the landfill are assigned to the second vehicle. This procedure is repeated until all of the trips have been assigned to vehicles. Selection

of trips in this manner provides a reasonable balance in route lengths by producing the shortest possible maximum distance traveled and the longest possible minimum-length route for each vehicle.

The pairing procedure is repeated for all trips of each vehicle capacity. If an odd number of sections are serviced by a vehicle of a particular capacity, the longest trip starting at the garage stands alone.

Subroutine PRSCHED is called once for each vehicle to produce a printed schedule of the route. Each segment in the trip produces a line of printed output in the schedule, unless a travel stretch is found on pieces of the same street having the same speed limits. Lines indicating lunch and break times are inserted in the schedule immediately following the description of the segment traversed during the lunch or break starting time. Unloading time at the landfill is also indicated by a separate line in the schedule.

The maps are plotted in three parts. The main program calls subroutine PATHPLT to plot the path traveled by the collection vehicle. Travel is indicated by a solid line, and collection is indicated by a dashed line. If collection is from one side of the street at a time, additional dashes are plotted perpendicular to the direction of travel. Subroutine PATHPLT counts the number of times each segment is traversed in each direction. The travel path is plotted on the correct side of the street, and subsequent traversals are drawn closer to the center of the street. Arrowheads indicating the direction of travel are appended to the path at approximately 3-inch intervals.

The main program calls subroutine STNAME to append street names to the map. STNAME obtains a street name and number and determines the length of space needed for the name. All of the segments are examined, and segment numbers and lengths for segments of the given street that are within the bounds of the map are saved. The segment lengths are sorted into decreasing order, with the segment numbers carried along during the sort. If the street name will fit on the longest untraveled segment, the name is plotted within the bounds of that segment. If the name will not fit in the longest untraveled segment, the longest segment is examined. If the name will fit there, it is plotted outside the segment. This procedure is repeated for each street name.

The main program calls subroutine MAPPLT to plot the sides of the streets. Each segment in the map description is examined. If the segment is at least partially within the map bounds, subroutine SHAPCOM is called with a 0.1-inch displacement from center indicated for the side of the street. Subroutine SHAPCOM adjusts the scale of the segment so that the side of the segment is plotted $\sqrt{2}/10$ (0.1414) inch shorter than the actual length of the segment. In this way, sides of street segments that meet at angles of 90 degrees or more are prevented from overlapping.

2. DATA STORAGE

Program PHASE4 obtains data from five sources: card input and files TAPE1, TAPE2, TAPE3, and TAPE9. Files TAPE1, TAPE2, and TAPE3 are the segment, node, and street-name data produced by program RCINPT. File TAPE9 is the route data produced by program PHASE3. A plot file, TAPE8, is generated by program PHASE4. The plot file contains maps of the routes.

The first executable statement in the main program reads the problem title, the refuse unit, and the consecutive-trip pairing indicator from the first two data cards. The title is stored in blank COMMON. The refuse unit is stored in array UNITS in COMMON block U. Storage for the pairing indicator is local to the main program. These values are not changed during the remainder of the program.

The second executable statement reads the segment data from file TAPE1. All of the data except variable AVMD are stored in blank COMMON. Storage for variable AVMD is in COMMON block MAPDATA. These data are not changed during the remainder of the program.

The third executable statement reads the node data from file TAPE2. Variables NHTOT and TOTREF are kept in storage local to the main program. The values are cleared just before the loop through statement 620, and the variables are reused in that loop. The remaining data, variable KNODES and arrays NODNUM, NBS, XNOD, and YNOD, are stored in COMMON block NODDATA and are not changed during the remainder of the program.

The main program calls subroutine STRIN to read the street-name data. The street names and numbers are buffered in from file TAPE3, or are read from the remaining cards in the first record if TAPE3 is empty. The names and numbers are stored in arrays in COMMON block STREETS and are not changed during the remainder of the program.

After control returns to the main program, two cards of time-restriction data are read. All of the data are stored in COMMON block TIMES. The stop time per house, stop time per unit of refuse, and unloading time at the landfill remain unchanged during the remainder of the program. The maximum trip time is changed to 4 hours if the value read is not greater than zero. The maximum route time is changed to 8 hours if the value read is not greater than zero. The starting time is changed to 8:00 o'clock if the value read is not greater than zero. The vehicle speed during collection is changed to 5 mph if the value read is not greater than zero. The duration and starting times for lunch and for each of the two breaks are not changed during the remainder of the program.

The main program reads vehicle-identification cards in the loop through statement 160. The values are kept in storage local to the main program and remain unchanged through the rest of the program.

The main program reads map-bounds cards in the loop through statement 190. Except for array NTRP, the values remain unchanged through the rest of the program. If the trip number is not greater than zero, it is reset to 1 at statement 185. Map bounds are computed in the main program between statements 450 and 480 for any trip that lacks a map-bounds card. All map-bounds variables are kept in storage local to the main program.

The part of the route data describing the path from the landfill to the garage is read at statement 270 in the main program. If no data are found on file TAPE9, the data are read from cards and are written to TAPE9. The remaining route data are read shortly before statement 400 of the main program. If the data come from cards, they are written to TAPE9. All of the variables are in storage local to the main program. The count of segments, N, and the section number, NSC, are reused later in the program. Values for vehicle capacity

and vehicle load, TRC and TRL, are updated each time path data are read. The path length, node numbers, segment numbers, and collection-or-travel indicators remain unchanged through the rest of the program.

The route data are reread from TAPE9 in the main program in the loop through statement 640. The trips are not printed in order of occurrence unless the consecutive-trip pairing indicator is greater than zero. Therefore, subroutine POSIT9 is used to position TAPE9 at the trip to be printed. If IPAIR is greater than zero, each pair of consecutive trips is assigned to one vehicle in the order in which the trips occur.

3. PURPOSE AND PERFORMANCE

In this section the simpler subroutines are described first so their workings will be understood when they are mentioned again in the descriptions of the more complicated subroutines. The main program is described last. Logic flowcharts are given in Appendix A. Complete program listings are provided in Appendix B. In Appendix C, the more important FORTRAN variables are described for each subroutine.

a. Function IFIND

Function IFIND uses a binary search to find a given number in an array and assigns the subscript of the number as the value of IFIND. If the number is not found, the function sets the value of IFIND equal to the negative of the subscript at which the number, to be in numerical order, should be inserted. (The array is assumed to be in increasing order.) The function has three arguments. Argument NUM is the number that is sought in array IARRAY. Argument IARRAY is the array to be searched. Argument LEN is the length of array IARRAY.

Function IFIND begins by checking to determine whether LEN > 0. If LEN ≤ 0, the function assigns a value of -1 to IFIND, and control returns to the calling program. The value -1 indicates that the number sought is not in the array and would be stored as the first entry in the array. The binary search uses variables II, IP, and IF as pointers. II is the subscript of the front

of the region being searched, IP is the subscript of the item being compared to the number being sought, and IF is the subscript of the last item in the region being searched. Variable II is set to 1 at statement 5. Variable IF is set to the length of the array. The pointer, IP, is the subscript midway between II and IF and is computed at statement 10.

Following statement 10, NUM is compared to IARRAY(IP). If NUM < IARRAY(IP), control transfers to statement 20, indicating that the number is in the front half of the region being searched. At statement 20 the final pointer is moved to the subscript preceding the point just searched. If NUM > IARRAY(IP), control transfers to statement 30, indicating that the number being sought follows the subscript just inspected. At statement 30 the initial pointer, II, is set to the present pointer, IP, plus 1. If the number sought is found at IARRAY(IP), control transfers to statement 50, where IFIND is set equal to the current pointer. Control returns to the calling program. Where NUM is unequal to IARRAY(IP), the initial or final pointer is moved and control resumes at statement 40. At statement 40 the final pointer is compared to the initial pointer. If IF ≥ II, control transfers to statement 10, where the search resumes on the appropriate half of the region examined previously. If the final pointer becomes less than the initial pointer, the number sought is not in the table. In this case, control resumes following statement 40, and the value of IFIND is set to the negative of the current pointer. If the number at the current pointer is less than the number being sought, IFIND is set to -(IP + 1) so the number can be inserted in the appropriate place. Control then returns to the calling program.

b. Subroutine STRIN

Subroutine STRIN reads the street numbers and names. STRIN first checks file TAPE3 for street data. These data would have been written by program RCINPT in the first phase of data entry. If the street data are found on file TAPE3, they are buffered in and stored in core. The street numbers, NUMSTR, and the street names, NAMSTR, are counted and totaled in variable NSTRS. The total number of street numbers and names read from TAPE3 is printed. If no street data are found on TAPE3, the street numbers and names are read from cards and stored. The number of street numbers and names read from cards is then printed. Control returns to the calling program.

c. Function HM

Function HM changes times of day that have been given in hours and fractions of hours to a character form of hours and minutes that will be used in the printout. Variable IH is the number of whole hours, and variable IM is the number of minutes. When the number of minutes in a given time of day is large enough to be rounded off to 60, IM is set to 0, and one whole hour is added to variable IH. An ENCODE statement is used to produce the character form of hours and minutes.

d. Subroutine NUMBER

Subroutine NUMBER appends numbers to plotted output. Its purpose is almost identical to that of the standard Calcomp number routine, the primary difference being that the last argument in subroutine NUMBER gives an alphanumeric format rather than an integer format code.

Subroutine NUMBER has six arguments. The first two give the coordinates, in plotter inches, of the lower left corner of the field. The third gives the height of the digits, in inches. The fourth is the number to be plotted. The fifth is the angle at which the number is to be plotted, measured in degrees counterclockwise from the horizontal. The last argument is an alphanumeric format up to 10 characters long, which describes the appearance of the plotted number.

Array TEXT is used to hold the character representation of the number. Up to 30 characters are allowed. The first executable FORTRAN statement sets this array to three words of blanks. The second statement moves the format into the second word of array FORM. The first and third words of this array have been preset to a left and a right parenthesis by a DATA statement. The ENCODE statement converts the number from binary form in variable NUM to character form in array TEXT, according to format FORM.

A character count, variable NC, is set to 30. The loop through statement 10 searches for the last nonblank character in array TEXT. Each time a blank is found, starting at the end of the TEXT array, the character count is

decremented by 1. When a nonblank character is encountered, control transfers to statement 20. Statement 20 calls the standard SYMBOL subroutine to plot the character representation of the number. Control then returns to the calling program.

e. Subroutine CUMTD

Subroutine CUMTD determines the total distance, time, number of houses, and refuse quantity for a given trip. Argument ISEG is the array of segments to be covered in a particular trip, CORT indicates whether collection or only travel is required on a particular segment, NSG is the number of segments, DIS is the total distance, TIM is the total time, NHT is the total number of houses, RQ is the total refuse quantity, SCOLL gives the vehicle speed in the collection area, TSTOPH is the stop time per household, and TSTOPR is the stop time per unit of refuse.

After the total number of houses, the total distance, the total time, and the total refuse quantity are set to 0, CUMTD checks each segment and adds its length to the distance accumulated to that point. CUMTD next checks the collection-or-travel indicator for that segment. If collection is required, three totals are changed: (1) TIM is increased by the time spent by the vehicle on that segment; the time is computed as the sum of the length of the segment divided by the speed of the vehicle when collecting refuse plus the estimated total collection time required for that segment. (2) NHT is increased by the number of houses on that segment. (3) RQ is increased by the amount of refuse for all of the houses on that segment. If no collection is required on the segment, only the total time is changed. In this case, the total time is increased by the length of the segment divided by the allowed speed of the vehicle for that segment. Control then returns to the calling program.

f. Subroutine SHLSRT

Subroutine SHLSRT sorts an array into either increasing or decreasing order. SHLSRT has four arguments: X, the array to be sorted; A, which is reordered as X is sorted so that each A always corresponds to the same X;

NW, the number of words to be sorted; and SGN, which indicates whether X is to be sorted in increasing or decreasing order.

The pointer separation, N, is set at half the number of words to be ordered. The pointer end value, K, is computed next. The initial and final pointers, J and L, are set. The values in arrays X and A at the inspection point are saved in variables XT and AT. SHLSRT then determines whether the final and the initial location values are in correct order. Variable SGN will have a value of 1 if X is to be sorted in increasing order, and a value of -1 if X is to be sorted in decreasing order. If the values are not in proper order, the initial value is moved to the final location. The final pointer is set equal to the initial pointer, and the initial pointer is moved up by the spacing value, N.

SHLSRT then determines whether the initial pointer has run off the front of the array. If it has, the saved values, XT and AT, are stored in the location designated by the final pointer. SHLSRT next checks the pointer spacing. If the spacing has not been reduced to 1, it is set equal to half the current value and the entire process is repeated. If the pointer spacing is equal to 1, control is returned to the calling program.

g. Subroutine POSIT9

Subroutine POSIT9 positions file TAPE9, the tape containing the route-description data, at the beginning of the trip to be processed. POSIT9 has two arguments: LSEQN is the sequence number of the last trip read from file TAPE9, and NSEQN is the sequence number of the trip to be processed next.

Subroutine POSIT9 first determines the position, on file TAPE9, of the next trip to be processed. If the next trip is the next record on the file, control is returned to the calling program. If the next trip is ahead of the last trip data read from the tape, TAPE9 is rewound and repositioned by means of a dummy read. If records must be skipped in order to reach the next trip data to be processed, a dummy read is used to find the correct position. Control is then returned to the calling program.

h. Subroutine SHAPCOM

Subroutine SHAPCOM sets up parameters in COMMON block COPARM that describe the geometric properties of a segment. These parameters are used by subroutine COORD to produce the coordinates of points on a segment.

Subroutine SHAPCOM has six arguments. Argument TOTLEN gives the total length of the segment, in miles. Argument AVMD gives the number of miles per map coordinate unit (MCU) on the overall map. W is the displacement from the center of the segment to the line drawn. DI and DF are displacements of the starting and ending points along the segment, in miles. Argument DIR gives the direction of travel. The values of the arguments are sent to subroutine SHAPCOM, and all output values from SHAPCOM are placed in COMMON block COPARM.

In COMMON block COPARM, variable SF indicates the shape of the segment. XNI and XNF are the x-coordinates of the initial and final nodes of the segment. YNI and YNF are the y-coordinates of these nodes. SX and SY are the slopes, in MCU per mile, in the x and y directions. RPR is the reciprocal of the radius of curvature for circular segments and the circular portions of S-curves. C11 and C12 are the position differences, in MCU, of the starting point and center of a circular arc or the first half of an S-curve. XCTR and YCTR are the center coordinates, in MCU, for a circular arc or half an S-curve. BR1 is the distance in miles from the start of a segment to some particular point on that segment. It is not used for straight segments. For circular segments, BR1 is the total perimeter. For an S-curve, BR1 is the perimeter to the midpoint of the S-curve. For a rectangular segment, BR1 is the distance to the first bend in the rectangle. For an angle, BR1 is the distance to the vertex. BR2 is defined only for rectangular segments and angles. For a rectangular segment, BR2 is the distance in miles from the beginning of the segment to the second bend. For an angle, BR2 is the length of the second side. A value of 1 is assigned to variable NPC. Arrays A, B, RATIO, XINT, and YINT are used in various combinations according to the segment shape being processed.

Subroutine SHAPCOM begins execution by assuming that the shape code indicates a straight line. BR1 and BR2 are set to 0. DX and DY, the x- and y-components of the vector from the initial node to the final node on the segment, are computed. The x- and y-components of the slope of the vector, measured

in MCU per mile, are computed and stored in SX and SY. Values are assigned to A(1), B(1), XINT(1), YINT(1), and RATIO(1). The shape code is tested; if the segment proves to be a straight line or is not to be plotted, the subroutine returns control to the calling program. For any other shape code, execution continues. The angle of the vector from the initial node to the final node is computed as variable THETA. The length of the vector, D, is computed in miles. The shape code is checked again. If it indicates a shape other than a circular arc or an S-curve, control transfers to statement 40. If it indicates a circular arc or an S-curve, variables XE and YE are set to the x and y values of the final node of the street segment. Other variables are set as SHAPCOM assumes a circular arc shape code. The shape code is then tested again. If an S-curve is indicated, variables XE and YE are reset to the coordinates of the midpoint of the S-curve. Break indicator BR1 is reset to the perimeter from the starting point to the center of the S-curve. Variable DD is set to half the distance from the starting point to the stopping point. Other variables necessary to process an S-curve street segment are set. If the shape code indicates either a circular arc or a portion of an S-curve with a left direction, the value of variable SGN is replaced by -SGN. Other necessary variables are set, and RPR, the reciprocal of the radius of curvature, is calculated. The approximate RPR is improved by a series of linear interpolations. When RPR is within the desired range of accuracy, the radius of curvature, R, is computed. A temporary variable, ARG, is evaluated. The height of the center of the circle from the line connecting the starting and stopping points is set to 0. If variable ARG is greater than 0, the height, H, is recomputed. The distance to the first break, BR1, is tested to determine whether the circular arc is greater than a half circle. If it is, the sign of the height is changed. The x- and y-coordinates of the center of the circle are computed. The value of RATIO(1) is based on the values of W and RPR. RATIO(2) is set at 2.0-RATIO(1). The components of the vector from the center to the starting point, C11 and C12, are computed. All variables needed to compute points on the circular arc or the S-curve are now available, so control returns to the calling program.

At statement 40 the shape code is tested to determine whether it indicates a rectangular segment. For the rectangular segment, the distance from the start to the first bend, BR1, is computed. If this distance is greater than 0.05 of the total length of the segment, control transfers to statement 50,

where the segment is treated as a rectangular shape. Otherwise, the rectangle is assumed to be so shallow that a straight-line approximation is adequate, and the shape code is set to 0. Control then returns to the calling program. At statement 50 the perimeter to the second bend in the rectangle, BR2, is computed. The values of all other variables, including all of arrays A, B, XINT, YINT, and RATIO, are set. Control is then returned to the calling program.

At this point, the only segments that remain to be processed are the angles. Both BR1, the length of the first side of the angle, and BR2, the length of the second side, are computed. A temporary variable, F, is computed. This variable and the total perimeter of the angle are used to compute the height of the vertex above the line connecting the starting and stopping nodes. The height, H, is then used in the computation of the x- and y-coordinates of the vertex of the angle. All other variables needed to compute points on the angular segment are computed. Control is returned to the calling program.

i. Subroutine COORD

Subroutine COORD is given a distance, in miles, from the beginning of a segment and returns the coordinates in MCU. Parameters describing the current segment are stored in COMMON block COPARM by subroutine SHAPCOM before COORD is called. Argument CUMLEN is the cumulative length, in miles. Arguments XX and YY are the coordinates returned for a point CUMLEN miles from the start of the segment. IERR will have a value of 1 or 0, indicating whether the values of the coordinates have been calculated in subroutine COORD. A value of 1 shows that an error has occurred and that the coordinates have not been calculated.

Variable S is set equal to the cumulative length. The shape code, SF, is checked. If it is not equal to 0, control transfers to statement 10. Otherwise, processing continues for a straight segment. The coordinates of the point on a straight-line segment are computed and stored in variables XX and YY. Control transfers to statement 80.

At statement 10, if the segment is not straight, the shape code is tested to determine whether the segment is a circular arc or an S-curve. If

it is neither, control transfers to statement 30. For circular and S-curve segments, the reciprocal of the radius of curvature is stored in RIP. The coordinates of the center of the circular portion are stored in XC and YC. The components of the vector from the center of the circle to the initial node are stored in C1 and C2. If the point on the segment is less than or equal to 0.999 of the first break distance or if the shape code indicates a circular segment, control transfers to statement 20. The statements following this test change parameters to generate coordinates for the second circular portion of an S-curve. The sign of the reciprocal of the radius of curvature is then reversed. The cumulative distance, S, is set to the distance from the midpoint of the S-curve. The coordinates of the center of the second portion of the S-curve, XC and YC, are computed. Variables C1 and C2 are recomputed for the new center. The sine and cosine of the angle subtended by the perimeter corresponding to S are computed. The coordinates of the point, XX and YY, are computed, and control transfers to statement 80.

At statement 30, the shape code is checked to determine whether the segment shape is a rectangle. If it is not, control transfers to statement 60. If it is, variable SGN is set to 1. If the shape code indicates a left rectangle, SGN is reset to -1. If S, the distance along the rectangle, is greater than 1.05 times the length of the first side, control transfers to statement 40 in subroutine COORD. If S is greater than 0.95 times the length of the first leg of the rectangle, S is set equal to the length of the first leg. The x- and y-coordinates of the point on the first leg are computed by linear interpolation, and control transfers to statement 80.

At statement 40, S is tested to determine whether it falls on the second leg of the rectangle. If S is greater than 1.05 times BR2, control transfers to statement 50. If S is greater than 0.95 times BR2, S is set equal to BR2. The x- and y-coordinates of the point on the second leg are computed by linear interpolation, and control transfers to statement 80.

At statement 50, the coordinates of a point on the third side of the rectangle are computed. Control transfers to statement 80.

At statement 60 the distance, S, is compared to the length of the first side of an angle segment. If S is greater than this length, control transfers to statement 70. If not, the x- and y-coordinates are computed by linear interpolation for a point on the first leg. Control transfers to statement 80. At statement 70 the distance along the angle is decreased by the length of the first leg of the angle. The coordinates of the point on the second leg are computed by linear interpolation.

At statement 80, a linear interpolation is performed to obtain, from the point on the segment, the coordinates of a point some distance from the center of the street. The distance is argument W to subroutine SHAPCOM. Control returns to the calling program.

j. Subroutine PRSCHED

Subroutine PRSCHED examines the trip data and prints the schedule for a particular trip. Argument NSC is the section number; ITRIP, the trip number; NTPS, the total number of trips per vehicle per day; NSP and NNP, arrays containing the numbers of the segments and nodes in the path; CORT, the collection-or-travel indicator; NSG, the number of segments on a particular trip; and TC, the vehicle capacity.

On the first trip for each vehicle, the route printing is initialized. If the current trip is the first trip of the day, PRSCHED will print out the necessary headings and the time, in minutes and hours, at which the truck leaves the garage. If the current trip is not the first trip of the day, PRSCHED will print out the time at which the truck leaves the landfill.

PRSCHED then begins processing the current segment. Function IFIND is used to find the street name and number. CORT is checked to determine whether collection or only travel is required on that segment. According to the value of CORT, PRSCHED begins either travel or collection processing. If collection processing is indicated, PRSCHED determines whether there is to be pick-up on only the right side or on both sides of the street. The totals for the number of houses, time, distance, and refuse amount are then accumulated. If only travel is indicated, the printed direction will be to drive on a particular street. Travel time and distance totals are accumulated.

Subroutine PRSCHED next checks for cross streets by using function IFIND on the node data. If there are cross streets, the street is broken up into street segments by the nodes that indicate street intersections. The direction to drive on a particular street, to pick up on the right side only, or to pick up on both sides of the street, is then printed out. At each street intersection, the time is checked to determine whether a morning, afternoon, or lunch break should have started while the vehicle was either driving on or collecting the last street segment. If so, the break is scheduled for the time at which the street segment is to be completed. The direction to break and the beginning and ending times for that particular break are printed out. PRSCHED next determines whether the final street segment for that trip has been completed. If so, the direction to unload is printed out, along with the starting and ending times for the unloading process. After the last piece of the trip has been processed, control returns to the calling program.

k. Subroutine STNAME

Subroutine STNAME appends street names to the maps of the collection areas. STNAME has four arguments: NSP, NNP, CORT, and NSG. NSP and NNP are arrays containing the street segment numbers and the node numbers of the travel path. CORT is the collection-or-travel indicator. NSG is the number of segments in a particular trip.

First, the map parameters are set. Then, a DECODE statement in STNAME changes internally the format of the stored street-name data. These data are examined, character by character. The street names are processed individually to find the first and last nonblank characters. The number of characters in the street name is determined and stored in variable NCH. Variable WIDTH is set originally to half the width of the street on the plotted map. The height of the letters to be printed is 1.6 times WIDTH; therefore, the street name will fit within the street boundaries. The width of the name, AWDTH, is a fractional part of the height of the letters times the number of letters.

A maximum of 20 segments with the same street name are saved temporarily. The nodes defining these segments are examined to determine whether the

segments are within map bounds. If the nodes show a segment to be within the boundary of the map, the segment number is saved in array NSGTEM, and the length of the segment is stored in array FLTEM. If the number of segments saved, NSV, is less than one, another street name is checked. Subroutine SHLSRT is called to sort the segments by length, with the longest segment first. STNAME then looks for an untraveled street segment long enough to hold the street name. If it finds one, the name will be printed inside the boundaries of that segment. Otherwise, the street name will be written outside the longest street segment (if it is longer than the street name).

The coordinates for printing the street name are set up. Subroutine SHAPCOM is called to ensure that the printing of the street name will follow the shape of the street segment. Next, the individual characters are plotted by calls to subroutines COORD and SYMBOL. Subroutine STNAME then determines whether any more street names must be processed. When the processing of all street names has been completed, control is returned to the calling program.

1. Subroutine MAPPLT

Subroutine MAPPLT draws a street map with double lines representing the sides of the streets. The subroutine has three arguments. The first, NRT, is the route number. The second, ITRIP, is the trip number. The third, NTPS, is the number of trips allowed per vehicle per day.

The coordinates of the region bounding the map are stored in COMMON block MAPDATA. In this COMMON block, variables XMIN and XMAX are the minimum and maximum x-coordinates for the map. XLEN is the length, in inches, of the map in the x-direction. YMIN, YMAX, and YLEN are the corresponding variables in the y-direction. YHCUT is the height, in plotter inches, at which the map must be sliced into strips. Variable AVMD contains the miles per MCU conversion factor for each map. WIDTH is half the width of the street, in plotter inches.

MAPPLT begins by retrieving or computing the map bounds, the height of a strip of the map (PHGT), the maximum length, the number of map strips (MX), the map-scale factors, and the interval, in MCU, at which the strips are to be cut.

The loop through statement 210 controls the plotting of each side of the street. Variable W is the width, in miles, from the center to the side of the street. The loop through statement 200 tests each segment to see whether it falls within the frame of the map; if it does, the segment will be plotted. Variables NI and NF are set equal to the numbers of the nodes bounding the segment. The midpoint coordinates of the segment are saved in variables XMD and YMD. The lines in the node-number array at which the initial and final nodes occur are saved in variables NS1 and NS2. The coordinates of each node are retrieved.

Initially the segment is assumed to be entirely within the bounds, and indicators INBI, INBM, and INBF are set to 1. If the coordinates of the initial node lie outside the frame of the map, INBI is set to 0. Similar tests are made on the coordinates of the midpoint of the segment and the coordinates of the final node of the segment. If all three points are outside the frame of the map, control transfers to statement 200 and the segment is not plotted. For segments that are at least partially within the frame of the map, the total length of the segment, in miles, is saved in variable TOTLEN. The number of points to be used in plotting half the segment, NPMID, is computed. The number will be restricted to a maximum of 10 points. The total number of points per segment, NPPSEG, is set to twice NPMID.

Subroutine SHAPCOM is called to set up the parameters needed to generate coordinates of points on the segment. The cumulative length along the segment is initially set to 0. A step size, DS, is computed as the total length of the segment divided by the number of points to be plotted. The coordinates of the initial node are stored in variables XX and YY. The number of the strip of the map into which the node falls is computed. Both a current value of the strip number, NMAP, and a value for the previous point, NMAPO, will be used. The pen position, up or down, is determined by whether the initial point is in bounds. Variable IPEN will be 3 if the point is out of bounds and 2 if the point is in bounds. If the point is out of bounds, control transfers to statement 130. If not, the coordinates of the point are converted to plotter inches and stored in variables XP and YP.

Statement 130 starts a loop through statement 170 that will advance the pen through the remaining points on the segment. The cumulative length is incremented by DS. Subroutine COORD is called to obtain the coordinates of the point in MCU.

At statement 140 the coordinates are converted to plotter inches. The point is assumed to be in bounds, and variable INB is set to 1. If the coordinates of the point are out of bounds, INB is reset to 0. If the pen has been up and the current point is out of bounds, or if the strip number is greater than the number of the final strip, control transfers to statement 160. Otherwise, the pen is moved to the position of the current point. If the pen is up, it is lowered. Variable IPEN is recomputed to reflect whether the point is in bounds.

At statement 160 the number of the current strip is computed. If the current strip number is equal to the previous strip number, control transfers to statement 170. If not, the old strip number, NMAPO, is set equal to the current strip number. IPEN is set to 3 to indicate that the pen is up. Control transfers to statement 140, where the coordinates of the current point on the new strip will be computed.

Statement 170 is the end of the loop that causes the segment to be drawn. Statement 200 is the end of the loop that draws one side of the various segments. Statement 210 marks the end of the loop on each side of the street.

Subroutine SYMBOL is called to append the problem title to the lower left corner of the map. A solid line is drawn 2 inches back from the lower right corner by two calls to PLOT. Subroutine SYMBOL appends the legend TRAVEL to the line.

The loop through statement 230 controls the plotting of two more lines and legends. Subroutine SYMBOL plots the legend COLLECT BOTH SIDES on the first pass through the loop and the legend COLLECT RIGHT SIDE on the second pass. The loop through statement 220 plots a horizontal dashed line. On the second pass through the 230 loop, small vertical lines are also plotted.

At statement 300 the plotter pen is positioned 2 inches beyond the end of the last map strip. Control returns to the calling program.

m. Subroutine PATHPLT

Subroutine PATHPLT draws the vehicle path, using a solid line to indicate travel and a dashed line to indicate collection. The subroutine has six arguments. The first two arguments, NSP and NNP, are arrays giving the segment and node numbers in the path. The third argument, CORT, is an array of collection-or-travel indicators. The fourth, NSG, is an array giving counts of segments in each of the four pieces of the trip. The fifth argument, NTRIP, is either 1 or 2 depending on whether the trip started at the garage or at the landfill. The sixth argument, NTPS, gives the maximum number of trips per day. The first three arguments are double-subscripted arrays. The first subscript corresponds to a step in the path; the second corresponds to a piece of the trip. The pieces are the path from the garage or landfill to the section; the path within the section; the path from the section to the landfill; and, for the final trip of the day, the path from the landfill back to the garage.

A counter, ILAST, is initially set to 0. The count of trip pieces, JF, is set to 4. If the current trip is not the last trip of the day, JF is reset to 3. JF is assigned the value 3 in the next statement. This statement causes the current version of the program to suppress the plotting of the trip from the landfill to the garage and can be deleted if the user wishes to show this part of the route.

The loop through statement 20 scans each piece of the trip. The number of segments in the piece is saved in variable N. The loop on statement 10 saves the segment numbers in arrays TRV and ISEG. The TRV array is sorted into increasing order by subroutine SHLSRT. The ISEG array is carried along.

The loop through statement 40 removes duplicate segment numbers from the ISEG array and closes up the empty spaces. Array ITRV is set to 0 in this loop.

The map bounds are retrieved from variables in COMMON block MAPDATA and are stored in variables XL, XR, YB, and YT. The maximum length, XMX; the number of map strips, MX; the map scale factors; and the intervals in MCU at which the strips are to be cut are computed.

The outer loop through statement 160 controls the scanning of the pieces of the trip. The inner loop through statement 160 will plot each path segment that falls within the frame of the map. Variables NI and NF are set equal to the numbers of the nodes bounding the path segment. The midpoint coordinates of the segment are saved in variables XMD and YMD. The lines in the node-number array at which the nodes occur are saved in variables LI and LF. If the nodes have been found in the NODNUM array, control transfers to statement 60. Otherwise, an error message is printed and control transfers to statement 160. At statement 60, the node numbers are compared with the numbers of the nodes bounding the segment in the segment data. If the numbers are equal, control transfers to statement 80. Otherwise, an error message is printed and control transfers to statement 160.

Following statement 80, the coordinates of the nodes are retrieved. The segment is assumed to be entirely within the bounds of the map, and indicators INBI, INBM, and INBF are set to 1. If the coordinates of the initial node lie outside the frame of the map, INBI is set to 0. Similar tests are made on the coordinates of the midpoint and final nodes of the segment. If all three points are outside the frame of the map, control transfers to statement 160 and no path is plotted for that segment. For segments that are at least partially within the frame, the total length of the segment, in miles, is stored in variable TOTLEN. If the segment is straight, TOTLEN is computed using the end-point coordinates and the map distance conversion factor. The number of 1/10-inch steps to the middle of the segment, NPMID, is computed. If NPMID is 0, it is reset to 1. The number of points per segment, NPPSEG, is one less than twice NPMID. Variable CUMLEN is set to 0. A step size, DS, is computed from the total length and the number of points per segment.

Variable ISH is set to either 1 or 8, depending on the direction of travel on the segment. Similarly, variable DIR is set to 1 or -1. Segment KK is found in the ISEG array, and the line number is stored in variable LTR. The

number of times the segment has been traversed, NTRV, is retrieved from the ITRV array. In the ITRV array, the octal units digit gives the number of traversals from starting to ending node, and the octal tens digit gives the number of traversals in the other direction. The distance of the path line from the center of the street, W, is computed on the basis of the number of previous traversals. The appropriate digit in the ITRV array is incremented by ISH. The collection-or-travel indicator is retrieved and stored in variable ACT. Logical variable RSO is computed to indicate whether collection is from the right side only.

Subroutine SHAPCOM is called to set up the parameters needed to generate coordinates of the points on the path. Subroutine COORD is called to obtain the coordinates of the starting point of the segment, in MCU. The number of the strip of the map, NMAP, is computed. The pen position, up or down, is determined by whether the initial point was in bounds. The computation for variable IPEN yields 3 if the point is out of bounds or 2 if the point is in bounds. If the point is out of bounds or if an error has been detected by subroutine COORD, control transfers to statement 100. Otherwise, the coordinates of the point are saved in variables XLAST and YLAST. The coordinates are converted to plotter inches and stored in variables XP and YP. If the current point is the first to be plotted or if the current strip number differs from the previous number, subroutine PLOT is called to raise the pen and move it to the current point. Variable IFIRST is set to 0. Variable NMAPO is set equal to the current strip number. The pen is moved to the current point in the down position.

Statement 100 starts a loop through statement 150 that will advance the pen through the remaining points on the segment. The cumulative length is incremented by DS. A count of points drawn, KTOT, is incremented by 1. Subroutine COORD is called to obtain the coordinates of the point in MCU. If COORD finds an error, control transfers to statement 150.

At statement 110 the coordinates of the previous point are saved in variables XLAST and YLAST. The coordinates of the current point are computed in plotter inches and stored in variables XP and YP. The point is assumed to be in bounds, and variable INB is set to 1. If the coordinates of the point

are out of bounds, INB is reset to 0. If the path is to indicate collection, control transfers to statement 120. Otherwise, if the pen is in the up position, it is lowered. Variable IPEN is recomputed to reflect whether the point is in bounds. If the count of points, KTOT, is a multiple of KARO (in the current version of the program KARO is equal to 30), control transfers to statement 125. If not, control transfers to statement 140.

At statement 120, IPEN is set to 2. If the loop index, K, is even, control transfers to statement 130. Otherwise, IPEN is set equal to 3. If KTOT modulo KARO is greater than 1, control transfers to statement 140. Otherwise, starting at statement 125, displacements are computed for drawing an arrowhead. The arrowhead is drawn by three calls to PLOT. Control transfers to statement 140.

At statement 130, if collection is not from the right side only, control transfers to statement 140. Otherwise, a small mark is drawn perpendicular to the direction of the path. At statement 140, the strip number of the point is computed. If the strip number equals the previous strip number, control transfers to statement 150. Otherwise, if KTOT modulo KARO is less than or equal to 1, KTOT is incremented by 2. NMAPO is set equal to the current strip number. IPEN is set equal to 3. Control returns to statement 110, where the pen will be positioned at the current point on the new strip.

Statement 150 marks the end of the loop that plots the points on the current path segment. Statement 160 marks the end of the two loops on the segments in the pieces of the travel path. Control returns to the calling program.

n. Program PHASE4

Program PHASE4 uses the data prepared by programs RCINPT and PHASE3 to select a refuse-collection schedule for a particular area, prints that schedule, and plots a map showing the collection routes. The file assignments include TAPE1, a binary file containing segment data; TAPE2, another binary file containing node data; TAPE3, street-name data buffered in for use; TAPE8, the Calcomp plot tape; TAPE9, the formatted path data; and TAPE5, the input file. COMMON blocks are set up and variables dimensioned. A title, the unit for the

measurement of refuse quantity, and an indicator that specifies whether the trips are to be paired are read from cards.

The segment data are then read from TAPE1. These data include the number of segments and the following specific information for each segment: NSTR, the street number; NN1, the initial node for the segment; NN2, the final node for the segment; FLEN, the length of the segment; NH, the number of houses on the segment; FMPH, the speed limit on the segment, in miles per hour; NWAY, which indicates whether the vehicle will be on a one-way or a two-way street; RQF, the refuse-quantity adjustment factor; XMID, the x-coordinate of the midpoint of the segment; YMID, the y-coordinate of the midpoint of the segment; and SF, the shape code. AVMD, the map distance conversion factor, is the last word on the file.

The node data are then read from file TAPE2. The data include NHTOT, the total number of houses; TOTREF, the total amount of refuse; KNODES, the total number of nodes; and the following specific data for each node: NODNUM, the number of the node; NBS, the numbers of segments bounding the node; XNOD, the x-coordinate of the node; and YNOD, the y-coordinate of the node.

The title is then printed, along with the count of street segments and the count of nodes. Subroutine STRIN is called to read and save the street-name information. The following time-limitation data are read from cards: TSTOPH, the time spent at each house; TSTOPR, the time required to collect each unit of refuse; TUNLD, the time required to unload the vehicle; TMXTR, the maximum time allowed for each vehicle trip; TMXDAY, the maximum time allowed for each vehicle to be in operation during a day; TSTART, the starting time of the work day; SCOLL, the speed of the vehicle during refuse collection; DLUNCH, the duration of the lunch break; TLUNCH, the time the lunch break should begin; array DBRK, the durations of the mid-morning and mid-afternoon breaks; and array TBRK, the times at which the mid-morning and mid-afternoon breaks should start. Some of the data are then printed out as a check on the input.

The maximum trip time, the maximum working time per day, the starting time for the day, and the vehicle speed during collection are then examined.

Each value read should be greater than zero. If the value of any of these variables is not greater than zero, PHASE4 will change it to a predetermined value. TMXTR will be set to 4 hours, TMXDAY to 8 hours, TSTART to 8.00 hours, and SCOLL to 5 miles per hour. The rest of the input data are then printed.

The vehicle capacities and identifications are read next. TC, the truck capacity, and VID, the vehicle identification, are read for up to ten truck capacities with five words per vehicle identification. After they have been read, the data are printed. The loop is executed up to 11 times, as necessary, to reach the end-of-record card terminating the data.

Program PHASE4 next checks for map-bounds cards. The data read from cards are NSCN, the section number; NTRP, the number of the trip; XMN, the minimum x-value for the map; XMX, the maximum x-value; XLN, the length in the x-direction; YMN, the minimum y-value; YMX, the maximum y-value; and YLN, the length in the y-direction. If no map bounds are specified for a particular map, the map will show travel in the collection region but not necessarily the path to or from the garage or landfill. If bounds are specified for any of the maps, the data are printed out.

The path data are read from TAPE9 for the travel from the landfill back to the garage. These data include N, the number of segments in the path; DIST, the distance covered; NNP, the node numbers; NSP, the segment numbers; CORT, the collection-or-travel indicators; and the final node number. If the path data are not found on TAPE9, PHASE4 will check for data on cards on input file TAPE5. If the data are on TAPE5, TAPE9 is rewound and the card images are written to file TAPE9. If no path data are found on TAPE9 or on cards, the job is terminated.

After the data files have been examined, plotting is initialized by calls to PLOTS and PLOT. The first call to CUMTD determines the time and distance from the landfill to the garage. The following variables are now set to 0: NSCOLD, the old section number; NTRIP, the trip number; TOTD, the total distance; TOTT, the total time; and TOTR, the total refuse collected. Data for travel to, within, and from the collection region are read. If the data are entered from cards (TAPE5), they are written to file TAPE9. The data consist

of N, the number of segments in the path; DIST, the distance covered; NSC, the section number; TRC, the truck capacity; TRL, the refuse load; NNP, the numbers of the nodes in the path; NSP, the numbers of the segments in the path; and CORT, the collection-or-travel indicators. A call to CUMTD provides the total time, distance covered, and refuse collected for each part of the trip. These totals are accumulated in variables TOTD (total distance), TOTT (total time), and TOTR (total refuse collected). If the part of the trip just completed was travel within the collection area, the number of houses serviced is stored in variable NHS.

PHASE4 next determines whether map-bounds cards have been read for the collection region just processed. If so, the program continues at statement 480. If not, XMAX, XMIN, YMAX, and YMIN are set to default values. These default values are determined by a consideration of both the nodes and the segments within the collection region. The section number and the initial values of XMAX, XMIN, YMAX, and YMIN are printed out. A scaling factor, SC, is determined, and the values are adjusted in preparation for plotting the map of the collection region. The adjusted values are then printed.

If path data are read from cards, an end-of-file mark is placed on TAPE9, and the file is rewound. The trip information, consisting of the section number, trip number, distance covered, time, number of households serviced, capacity, and load, is printed.

If the trips are not to be paired in the order in which they occur, subroutine SHLSRT is called to sort the vehicles by capacity. Pointers to the trips in the IORD array are carried along during the sort.

In the loop through statement 540, the travel times for each vehicle capacity are sorted. If the longest time exceeds the maximum trip time specified by the user, a warning message is printed and the maximum trip time is extended. The maximum trip time for vehicles of each capacity is saved in the TMXTRV array.

A page heading for the final route summary is printed. The loop through statement 620 accumulates the values printed in the route summary. The

loop through statement 550 seeks the line number of the vehicle identification corresponding to the vehicle capacity that is now being processed by loop 620.

The loop through statement 570 examines the trips selected for vehicles of the capacity now being processed. The distance saved by using the second rather than the first trip choice for each section is stored in array TEM. If the time for the first trip choice exceeds the maximum trip time, the distance saving is increased by 2000. If the trip time for the second choice exceeds the maximum trip time, the distance saving is decreased by 2000. This adjustment is made so that when one trip choice exceeds the maximum time and the other does not, the program will select the trip that takes less time.

Subroutine SHLSRT is called to sort the distance savings into increasing order. The loop on statement 585 replaces the distance savings in array TEM by the time for the appropriate trips. If more than one route is present and if there are two choices of trip per section, subroutine SHLSRT is called to sort the times of the first trips into increasing order. The trip pointers are carried along during the sort. If there is more than one afternoon trip, subroutine SHLSRT is called to sort the afternoon trip times into increasing order. The trip pointers are carried along during the sort.

The loop through statement 610 accumulates the total number of houses, refuse quantity, time, and distance for each route. These values are printed as part of the route summary. The section numbers of the trips comprising the route are saved in the IRS array. Cumulative values of distance, time, refuse, and houses serviced are also accumulated in the loop. Following statement 620, the totals for all routes are printed.

File TAPE9, which contains the route data, is rewound. The sequence number of the last trip read, LSEQN, is set to -1 to indicate that the file is positioned at the beginning of the data. The trip from the landfill to the garage has sequence number 0. The trips servicing each section have sequence numbers 1 through the number of trips.

The loop through statement 680 generates the maps and schedule for each route. The loop is executed one time more than the number of routes so

that maps without paths can also be plotted. Variable JTRIPS is computed to be the number of trips per route. The loop through statement 670 controls schedule and map generation for each trip. The section number is obtained from the IRS array. The sequence number for the trip, ISEQN, is computed. Subroutine POSIT9 is called to position TAPE9 at the beginning of the desired trip. The loop through statement 640 reads the three pieces of the trip from TAPE9.

If the trip is the first trip of the route, the route number, the problem title, and the vehicle identification are printed as the heading for the schedule. Subroutine PRSCHED is called to print the schedule for the trip.

The loop through statement 660 examines the map-bounds data. The bounds and lengths of the map are retrieved. If the bounds in the x or y directions allow no width or height to the map, the map plotting is bypassed and control transfers to statement 670. If the loop index indicates a route with trips in it, subroutine PATHPLT is called to plot the path. Subroutines NUMBER and SYMBOL are called to append a legend to the map. Subroutine STNAME is called to append street names to the map. Subroutine MAPPLT is called to draw the sides of the street on the map.

Statements 660, 670, and 680 mark the ends of their respective loops.

Subroutine PLOT is called to terminate the plot file. Subroutine EXIT is called to return control to the system.

(The reverse of this page is blank.)

SECTION IV
INPUT AND OUTPUT

1. INPUT

Input to program PHASE4 consists of card input and four disk files. Three of the disk files are generated by program RCINPT and one by program PHASE3. The files are read by the main program, PHASE4, and by subroutine STRIN.

a. Card Input

The form and contents of the data cards are shown in Table 1. Four types of data cards are required for program PHASE4; three more types are optional. The required cards are a title card, a refuse-unit card, two time-restriction cards, and vehicle-identification cards. The optional cards are map-bounds, street-name, and path-information cards.

Data cards for Kirtland Air Force Base are shown in Appendix D. A record of map-bounds cards is included. In this case, there are no street-name or path-information cards because these data are obtained from disk files. A detailed description of the preparation of these data cards can be found in Reference 1.

b. Disk Files

Disk file TAPE1 contains segment data and the map distance conversion factor (miles per MCU) for the overall map. All of the data are read by one binary READ statement. The first word is the count of the segments. The segment data follow, 11 words per segment. After the segment data comes the overall distance conversion factor. The following list is used in the READ statement:

Reference

[1] Iuzzolino, Harold J., *Air Force Refuse-Collection Scheduling Program*, CEEDO-TR-77-32, Tyndall Air Force Base, Florida, January 1978.

TABLE 1. PHASE4 DATA CARDS

Record	Card	Columns	Format	Contents
1	1	1-80	8A10	Title
	2	1-20	2A10	Refuse units
		21-25	I5	Trip pairing control
The following card is optional and may be repeated up to 300 times.				
	3	1-5	I5	Street number (right justified)
		11-60	5A10	Street name (left justified)
End of Record				7-8-9 multipunched in column 1.
If card 3 is omitted, omit End of Record and append the next three cards to record 1.				
2	1	1-10	F10.0	Stop time per household, in minutes
		11-20	F10.0	Stop time per unit refuse, in minutes
		21-30	F10.0	Unloading time, in minutes
		31-40	F10.0	Maximum trip time, in hours
		41-50	F10.0	Maximum working time per day, in hours
		51-60	F10.0	Starting time, in hours
		61-70	F10.0	Average vehicle speed, in mph, between two collection points on a street
	2	1-10	F10.0	Duration of lunch, in minutes
		11-20	F10.0	Starting time of lunch, in hours
		21-30	F10.0	Duration of first break, in minutes
		31-40	F10.0	Starting time of first break, in hours
		41-50	F10.0	Duration of second break, in minutes
		51-60	F10.0	Starting time of second break, in hours
The following card may be repeated up to 10 times.				
	3	1-10	F10.2	Vehicle capacity
		11-60	5A10	Vehicle identification
End of Record				7-8-9 multipunched in column 1.

TABLE 1. PHASE4 DATA CARDS (Concluded)

Record	Card	Columns	Format	Contents
The following card is optional and may be repeated up to 100 times.				
3	1	1-5	I5	Section number
		6-10	I5	AM/PM trip indicator
		11-20	F10.0	Minimum x-coordinate
		21-30	F10.0	Maximum x-coordinate
		31-40	F10.0	Length of x-direction, in inches
		41-50	F10.0	Minimum y-coordinate
		51-60	F10.0	Maximum y-coordinate
		61-70	F10.0	Length of y-direction, in inches
End of Record				7-8-9 multipunched in column 1.
The following record is used only if path data are not on file TAPE9.				
4	1	1-5	I5	Count of segments in this piece
		6-10	F10.3	Total distance for this piece of the trip, in miles
		11-15	I5	Section number
		16-25	F10.3	Vehicle capacity
		26-35	F10.3	Total refuse quantity for this trip
	2	1-5	I5	Node number
		6-9	I4	Segment number
		10	A1	Collection-or-travel indicator
The above three items may be repeated to the end of the card. This card is repeated until the entire path to the collection region has been described.				
The two cards are used twice more for each trip, once to describe the path in the collection region and again to describe the path from the collection region to the landfill.				
End of Record				7-8-9 multipunched in column 1.

NSEG, (NSTR(I), NN1(I), NN2(I), FLEN(I), NH(I), FMPH(I), NWAY(I), RQF(I), XMID(I), YMID(I), SF(I), I = 1, NSEG), AVMD

The items in the list are defined as follows:

NSEG = number of segments
NSTR(I) = street number
NN1(I) = starting node number
NN2(I) = ending node number
FLEN(I) = street length, in miles
NH(I) = number of houses
FMPH(I) = speed limit, in mph
NWAY(I) = number of ways of travel
RQF(I) = refuse-quantity adjustment factor
XMID(I) = x-coordinate of segment midpoint
YMID(I) = y-coordinate of segment midpoint
SF(I) = shape code
AVMD = map distance conversion factor, in miles per MCU

- These are repeated for each segment.

The sign of the number of houses is used to indicate whether collection is on both sides or on only the right side of the segment. NH is negative to indicate collection on only the right side of the street, or positive to indicate collection from both sides of the street.

Disk file TAPE2 contains refuse-quantity information and the node data. All of the data are read by one binary READ statement. The first three words are the total number of houses or stops, the total refuse quantity, and a count of the nodes. The node data follow, four words per node. The following list is used in the READ statement:

NHTOT, TOTREF, KNODES, (NODNUM(I), NBS(I), XNOD(I), YNOD(I), I = 1, KNODES)

The items in the list are defined as follows:

NHTOT = total number of houses or stops
TOTREF = total refuse quantity
KNODES = count of nodes

NODNUM(I) = node number
NBS(I) = packed bounding-segment numbers
XNOD(I) = x-coordinate of node
YNOD(I) = y-coordinate of node

These are repeated for each node.

The index I corresponds to the Ith node. Variable NBS(I) contains up to 6 segment numbers, each occupying 10 of the 60 bits, for segments bounding node NODNUM(I). Since node numbers are assigned by the user, NODNUM(I) usually is not the same as I.

Disk file TAPE3 contains the street numbers and names (70 characters each). The data are read 100 streets at a time, using a BUFFER IN statement. Up to 300 streets may be read. The data consist of all of array NUMSTR (street numbers), followed by all of array NAMSTR (street names). Because the arrays are adjacent in storage, each record appears as follows:

NUMSTR(1) = number of first street
:
NUMSTR(100) = number of 100th street
NAMSTR(1,1)
:
NAMSTR(7,1)
= 7-word name of first street
NAMSTR(1,2) = first word of second street name
:
NAMSTR(7,100) = last word of 100th street name

The last record contains zeros in unused words.

Disk file TAPE9 contains the path data. A description of the path from the landfill to the garage occurs at the beginning of the file. Two trips follow for each section, one starting at the garage and one starting at the landfill. Each trip is divided into three pieces, and a header precedes each piece.

The path from the landfill to the garage and its header are read at statement 270 in the main program using a formatted READ statement. The remaining trips and their headers are read in the loop through statement 420 in the main program. The following list is used in reading the trip pieces:

N,DIST(J),NSC,TRC,TRL,(NNP(K,J),NSP(K,J),CORT(K,J),K=1,N),NNP(N+1,J)

The subscript J has values 1, 2, or 3 for the pieces of the trips servicing sections and value 4 for the trip from the landfill to the garage. The items in the list are

N	= Count of segments in piece	These data comprise the header.
DIST(J)	= Length of piece, in miles	
NSC	= Section number	
TRC	= Vehicle capacity	
TRL	= Vehicle load	
NNP(K,J)	= Node number	These data are repeated for each segment in the path.
NSP(K,J)	= Segment number	
CORT(K,J)	= Collection-or-travel indicator	
NNP(N+1,J)	= Number of the node terminating the trip	

The trip from the landfill to the garage lacks items NSC, TRC, and TRL in its header. The format for its header is I5,F10.0. The format used for the remaining headers and path data is I5,F10.3,I5,2F10.3/8(I5,I4,A1). The route data are read from either TAPE9 or cards. If the data are on cards, they are written to TAPE9 in the above format.

TAPE9 is reread in the loop through statement 640 in the main program. At this time the data are used to generate the schedules and maps.

2. OUTPUT

a. Plot File

File TAPE8, the plot file, will be on disk or tape depending on the procedure used by the local installation to produce plots. Each map occupies one file.

The final maps show travel to the collection region, within the collection region, and from the collection region to the landfill. Street names are appended to the map, and the direction of travel is shown by arrowheads on

the lines indicating collection or travel. Subsequent traversals along the same street segment are indicated by path lines drawn closer to the center of the street. Maps for Kirtland Air Force Base are given in Appendix E. The maps were produced using the map-bounds cards in Appendix D and have been reduced to half size.

b. Printed Output

The printed output consists of five sections: a listing of input data, map-bounds listings, a trip summary, a final route summary, and detailed descriptions of the routes.

The listing of input data is provided for verification of the card input. The problem title is printed first. Counts of the segments, nodes, and cards containing street-names are printed. Also listed are the unit of refuse and the time restrictions.

If bounds were specified for any of the maps, these data are printed on the second page of output. The third page lists map bounds selected by the program for sections not given map bounds by the user. The column headings are printed even if the program selects no bounds.

The trip summary gives general information about the trip choices for each section.

A final route summary is printed on the fifth page of output. The vehicle identification, vehicle capacity, section numbers of the trips, distance, time, households serviced, and refuse quantity are listed for each route. Totals are given for the distance, time, households serviced, and refuse quantity for all the routes.

The detailed route description shows the action to be taken on each street of each route. The first action printed instructs the driver to leave the garage. Then instructions are given either to drive on or collect from the street up to a particular cross street. The speed limit, distance traveled, and number of households serviced on that street are given. An estimate of the

of day at which the action is to be completed is given. If collection is ormed, the percent of vehicle capacity used is printed. After the descrip- of collection in the region, the path to the landfill is described. At end of the description of the day's final unloading of the vehicle, the h back to the garage is described.

Complete printed output for Kirtland Air Force Base is given in ppendix F.

time of day at which the action is to be completed is given. If collection is performed, the percent of vehicle capacity used is printed. After the description of collection in the region, the path to the landfill is described. At the end of the description of the day's final unloading of the vehicle, the path back to the garage is described.

Complete printed output for Kirtland Air Force Base is given in Appendix F.

SECTION V
PROGRAM REQUIREMENTS

1. SYSTEM

Program PHASE4 is written entirely in FORTRAN IV. The program runs on a CDC 6600 computer using a SCOPE 3.4.4 operating system.

Ten obvious types of computer-dependent coding occur in program PHASE4 and its subroutines. Subroutine PRSCHED assumes a 130-character output line. A 60-bit word is assumed in subroutine NUMBER. Subroutine PRSCHED assumes ten characters per word. System subroutine SHIFT is used by subroutines NUMBER and PRSCHED. An ENCODE statement is used in subroutine NUMBER and function HM; a DECODE statement is used in subroutine STNAME. The AND masking operation is used in subroutines NUMBER, PRSCHED, and PATHPLT. Asterisk-bounded text is used in format statements in the main program and subroutines PATHPLT, PRSCHED, and STRIN. Multiple replacement statements occur in the main program and in subroutines MAPPLT, SHAPCOM, CUMTD, and NUMBER. A dollar sign is used to separate FORTRAN statements in the main program and in all subprograms except function IFIND and subroutines STRIN and NUMBER. An R-format text variable is used in subroutines SHAPCOM and COORD.

More subtle types of machine dependencies may exist, according to the machine used.

2. STORAGE

The core requirement is slightly less than $111{,}000_8$ words. If data are written to file TAPE9, the maximum peripheral storage used by the file will not exceed 33,800 words. The maximum storage required by the plot file, TAPE8, should not exceed 30,000,000 words, although an estimate of 50,000 words per map is more typical.

3. TIME

The running time for PHASE4 varies with the number of maps and the number of segments per map. Since the number of segments per map can vary considerably, the following figures are estimates:

CP time = 2 seconds + 2.5 seconds x (number of maps)

I/O time = 2 seconds + 2.5 seconds x (number of maps)

PP time = 30 seconds + 3 seconds x (number of maps)

SECTION VI
PROGRAM LIMITATIONS

Program PHASE4 requires that the number of vehicle-identification cards be from one to 10 because of the array dimensions. The number of map-bounds cards may range from 0 to 100, but the number of sections without bounds cards plus the number of bounds cards must not exceed 100. Array dimensions impose this limitation, also.

The plotter may be either a flatbed or a drum plotter with a height of at least 30 inches. If a smaller drum plotter is used, the current assignment YHCUT=30., which appears shortly after statement 330 in the main program, should be replaced by a statement assigning to YHCUT the value of the drum height, in inches.

The use of a colon, a nonstandard CDC character, in the printing of time imposes a restriction on the storage and transmittal of the FORTRAN listing of the program and the route schedule. The colon will survive as a colon only on disk or cards. If a route schedule or the card images of the program are transferred to magnetic tape, the colon will be converted to a zero. On some devices, such as a microfiche unit, the colon may be completely deleted. One colon is used on each of the cards with serial numbers PHS42700, HM001100, and PRSC0220. It will be necessary to reinsert the colon on these cards if the cards are transferred to tape.

(The reverse of this page is blank.)

SECTION VII

ERROR MESSAGES AND CORRECTIVE ACTION

1. NO MAP BOUNDS WERE GIVEN FOR ANY TRIP.

EACH MAP WILL SHOW TRAVEL IN THE COLLECTION REGION BUT NOT NECESSARILY THE PATH TO OR FROM THE GARAGE OR LANDFILL.

Type: Warning.

Source: Program PHASE4.

Location: Where map-bounds listing would normally appear.

Meaning: Self-explanatory.

Action: If maps including travel to or from the garage or landfill are desired, use map-bounds cards.

2. WHERE NO BOUNDS WERE SPECIFIED, THE MAP WILL SHOW TRAVEL IN THE COLLECTION REGION BUT NOT NECESSARILY THE PATH TO OR FROM THE GARAGE OR LANDFILL.

Type: Informative.

Source: Program PHASE4.

Location: At the end of the map-bounds listing.

Meaning: Self-explanatory.

Action: If travel to or from the garage or landfill is desired on all maps, use map-bounds cards.

3. NO PATH DATA FOUND ON UNIT 9 OR ON CARDS.

JOB TERMINATED.

Type: Fatal.

Source: Program PHASE4.

Location: After map-bounds listing.

Meaning: No path data were found on cards or on file TAPE9.

Action: If the path data should be on TAPE9, see whether an ATTACH or REQUEST card for that file is present and correct. If path data should be on cards, look for an extra end-of-record card or an end-of-file card before the data. The deck may be missing.

4. THE MAXIMUM TRIP TIME WILL BE EXTENDED TO hh.hh HOURS FOR VEHICLES OF CAPACITY vvvvv.v

IF THIS IS UNSATISFACTORY, RERUN PROGRAM PHASE2 and PHASE3 WITH A SMALLER TIME LIMIT IN PHASE2.

Note: hh.hh represents a time in decimal hours.

vvvvv.v represents a vehicle capacity.

Type: Warning.

Source: Program PHASE4.

Location: Following the printing of the trip descriptions.

Meaning: A vehicle with the indicated capacity requires more than the user-specified maximum trip time. The program will automatically extend the maximum trip time to allow longer trips.

Action: If the time extension is not acceptable, rerun programs PHASE2 and PHASE3, using a smaller time limit in PHASE2.

5. PIECE p, nnnTH NODE, NUMBER mmmmm, IS INCORRECT.

Note: p is either 1, 2, or 3.

nnn is a sequence number from 1 through 101.

mmmmm is a node number from 1 through 99999.

Type: Warning, but the schedule will be incorrect.

Source: Subroutine PATHPLT.

Location: Following the printing of the schedule.

Meaning: Node number mmmmm, used in the path description on cards or on file TAPE9, is not present in the node data from file TAPE2.

Action: Verify that file TAPE2 and the path data on TAPE9 or cards correspond to the same problem. If the path is on cards, check for a card-punch error. If the total number of steps for the three pieces exceeds 200, the problem originates in program PHASE3. The last two paragraphs in Volume III, Section VII, of this report describe the cause of the error and possible corrective actions.

6. PIECE p, nnnTH SEGMENT, NUMBER mmmm, DOES NOT CONNECT TO A BOUNDING NODE.

Note: p is either 1, 2, or 3.

nnn is a sequence number from 1 through 100.

mmmm is a segment number from 1 through 1023.

Type: Warning, but the schedule will be incorrect.

Source: Subroutine PATHPLT.

Location: Following the printing of the schedule.

Meaning: One or both nodes bounding segment mmmm in a path description do not bound the segment in the data from file TAPE1.

Action: Verify that file TAPE1 and the path data on TAPE9 or cards correspond to the same problem. If the path is on cards, check for a card-punch error. If the total number of steps for the three pieces exceeds 200, the problem originates in program PHASE3. The last two paragraphs in Volume III, Section VII, of this report describe the cause of the error and possible corrective actions.

7. NO VEHICLE SPECIFIED FOR THIS CAPACITY.

Type: Warning, but the schedule may be incorrect.

Source: Main program PHASE4.

Location: In the route summary and in the heading for the schedule.

Meaning: A vehicle capacity from the route data on either TAPE9 or cards is not included in a vehicle-identification card.

Action: Check the vehicle-identification cards. A vehicle capacity may be missing or mispunched.

(The reverse of this page is blank.)

SECTION VIII
RECOMMENDED PROGRAM CHANGES

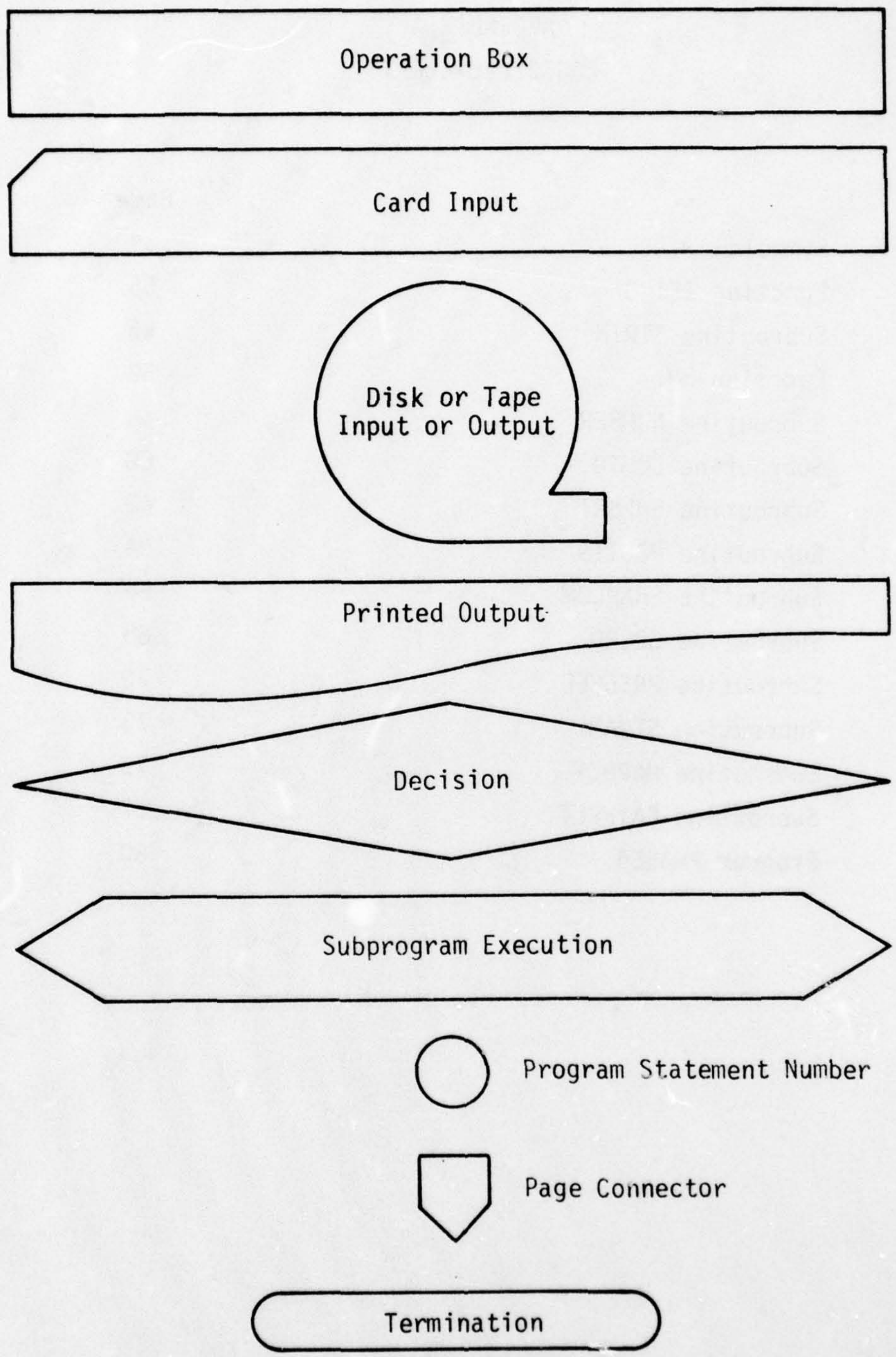
Two changes in program PHASE4 are recommended. The trip-pairing algorithm in the loop through statement 620 in the main program could be modified to generate single-trip routes when a two-trip route would exceed the maximum working time per day (TMXDAY) specified by the user. The change would require moderate modifications of the coding within the loop, and may require minor modifications in the path-, map-, and schedule-generation subroutines.

At present, the sides of the streets are plotted shorter than the segment length so that they meet only where streets come together at right angles. The fourth and fifth arguments to subroutine SHAPCOM allow length modifications at the initial and final ends of the segment. Additional coding could be inserted before the call to SHAPCOM in subroutine MAPPLT to adjust street lengths to take into account the angle between the segments. The change would require a knowledge of geometry, trigonometry, and, possibly, vectors.

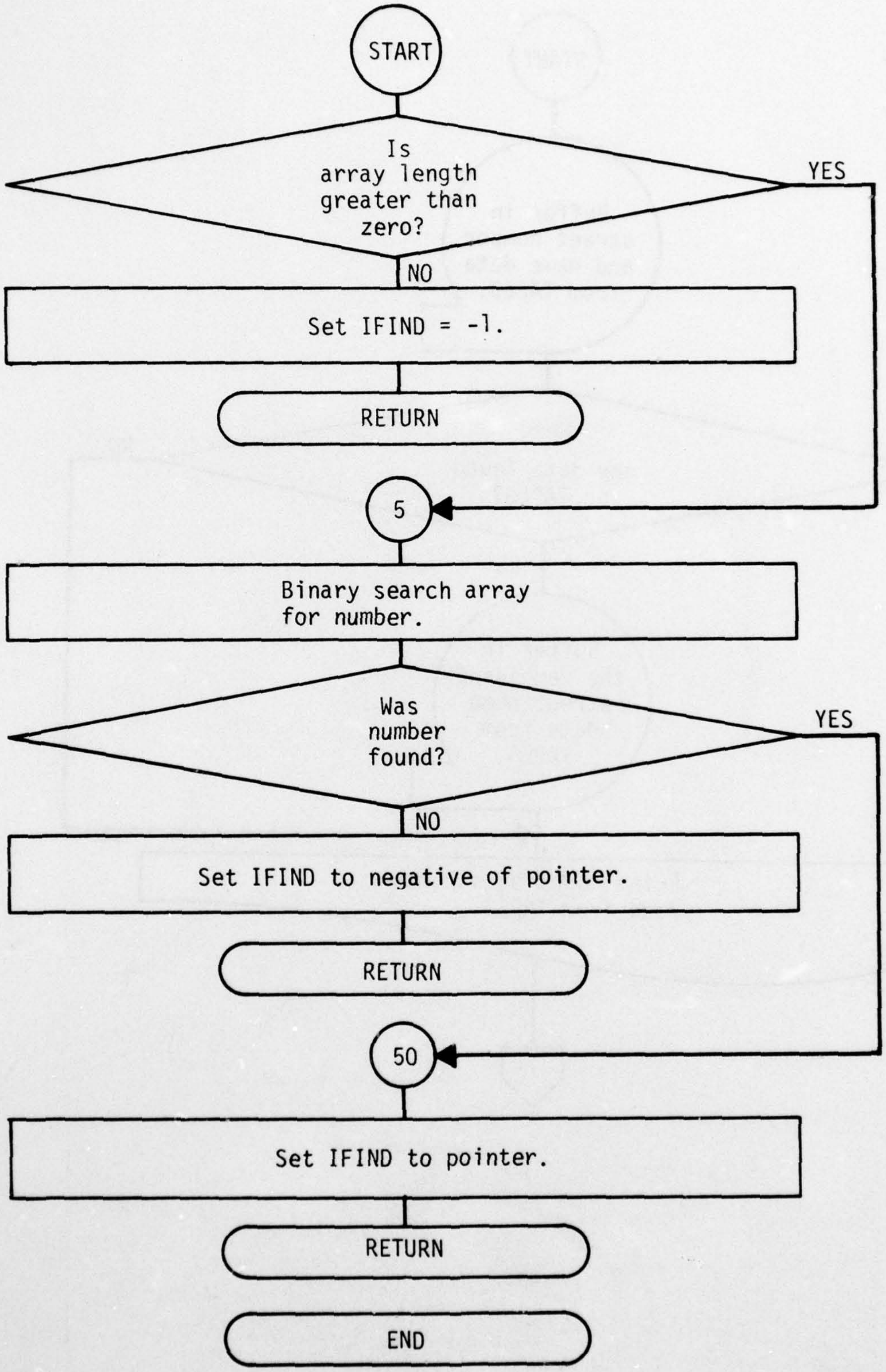
(The reverse of this page is blank.)

APPENDIX A

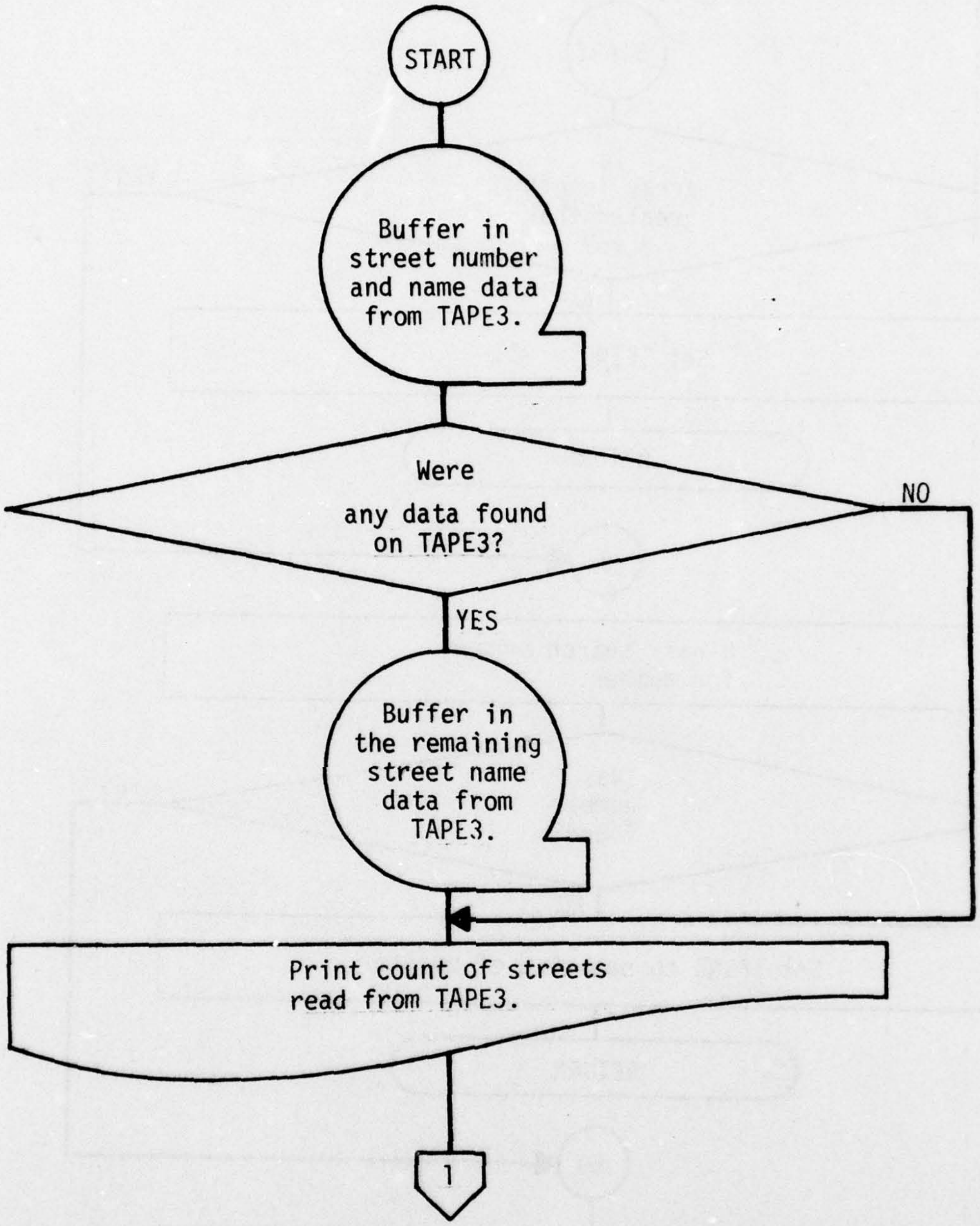
LOGIC FLOWCHARTS



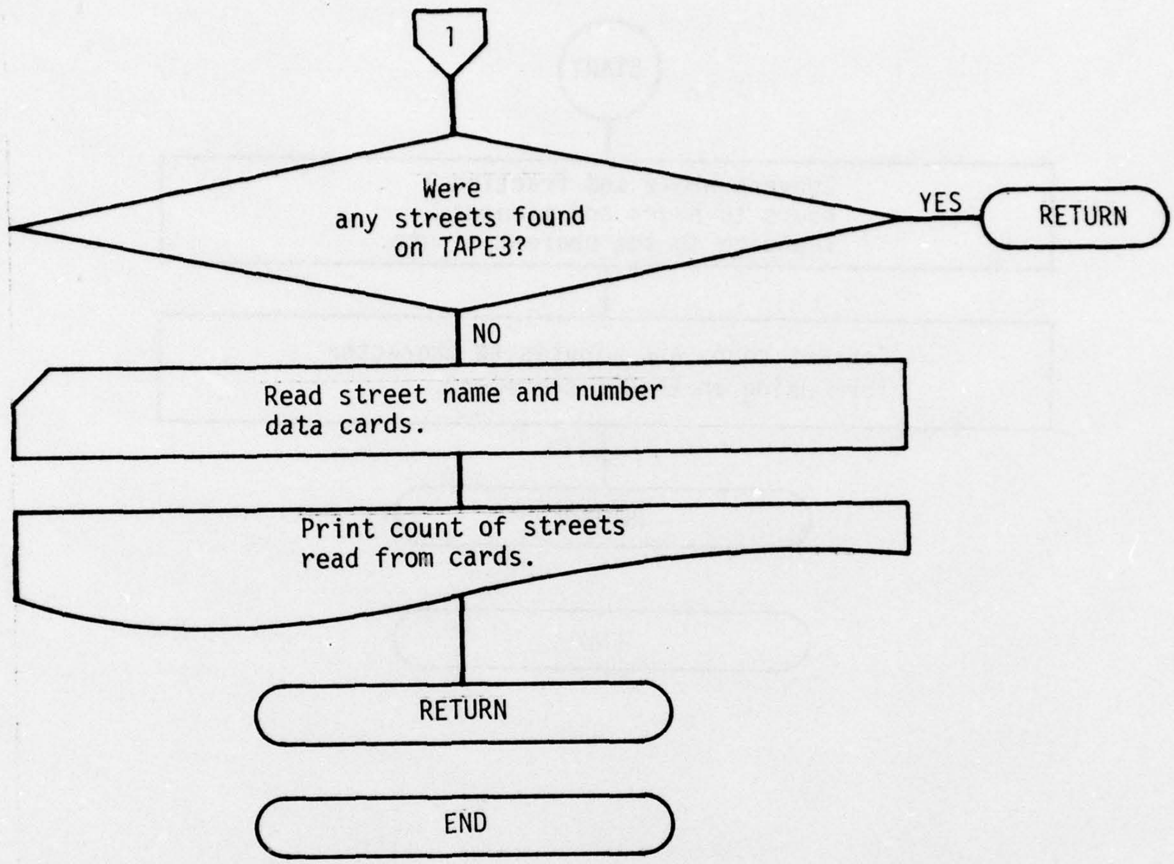
PROGRAM FLOWCHART SYMBOLS



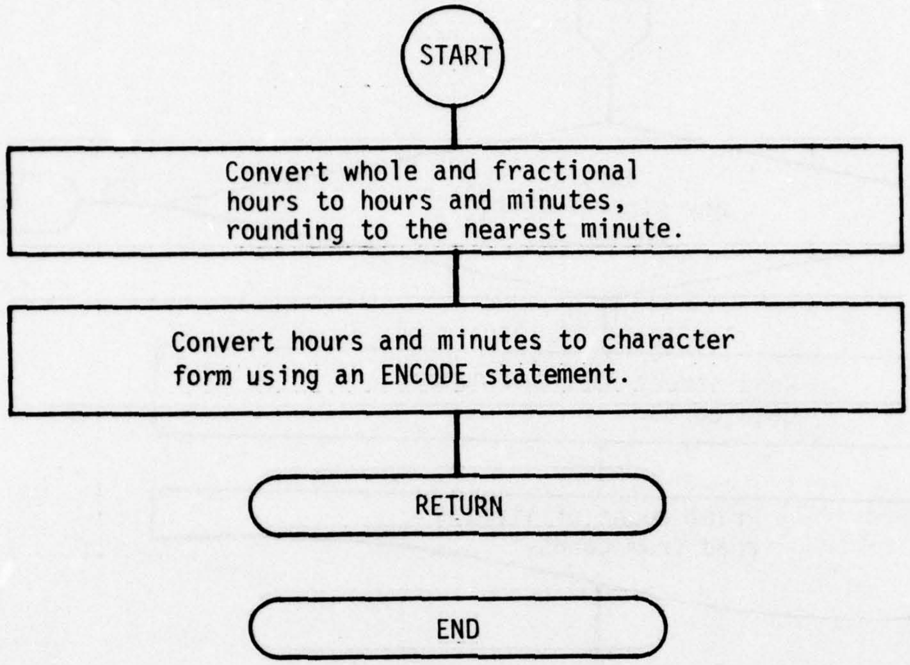
Function IFIND



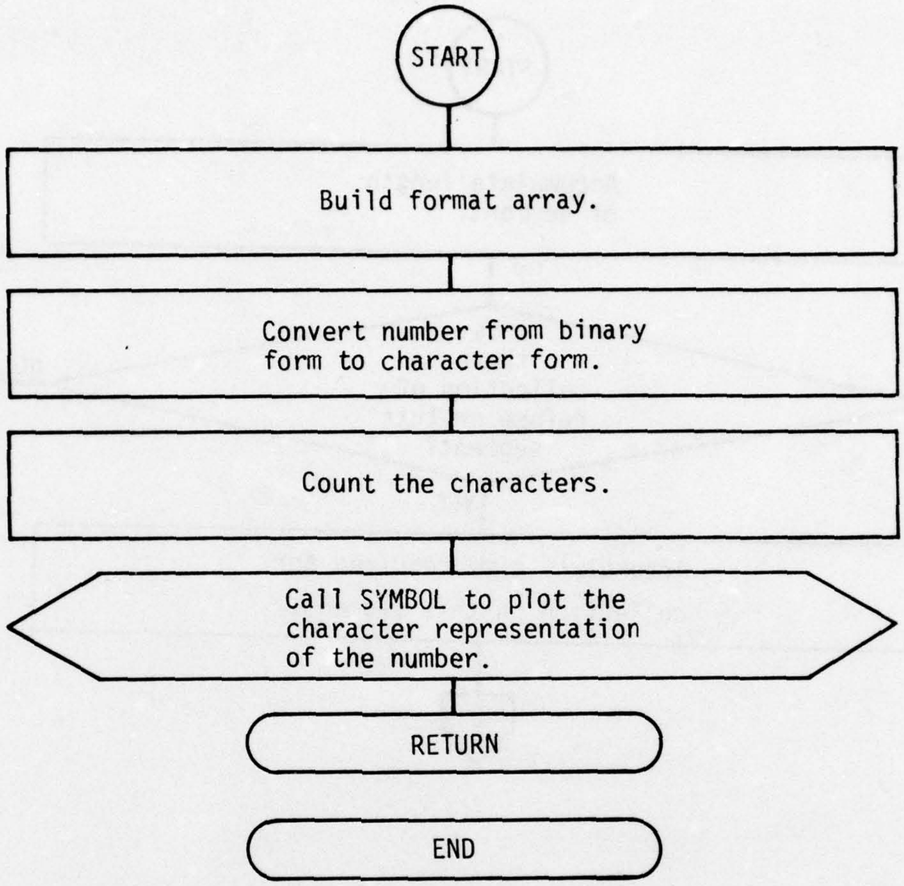
Subroutine STRIN



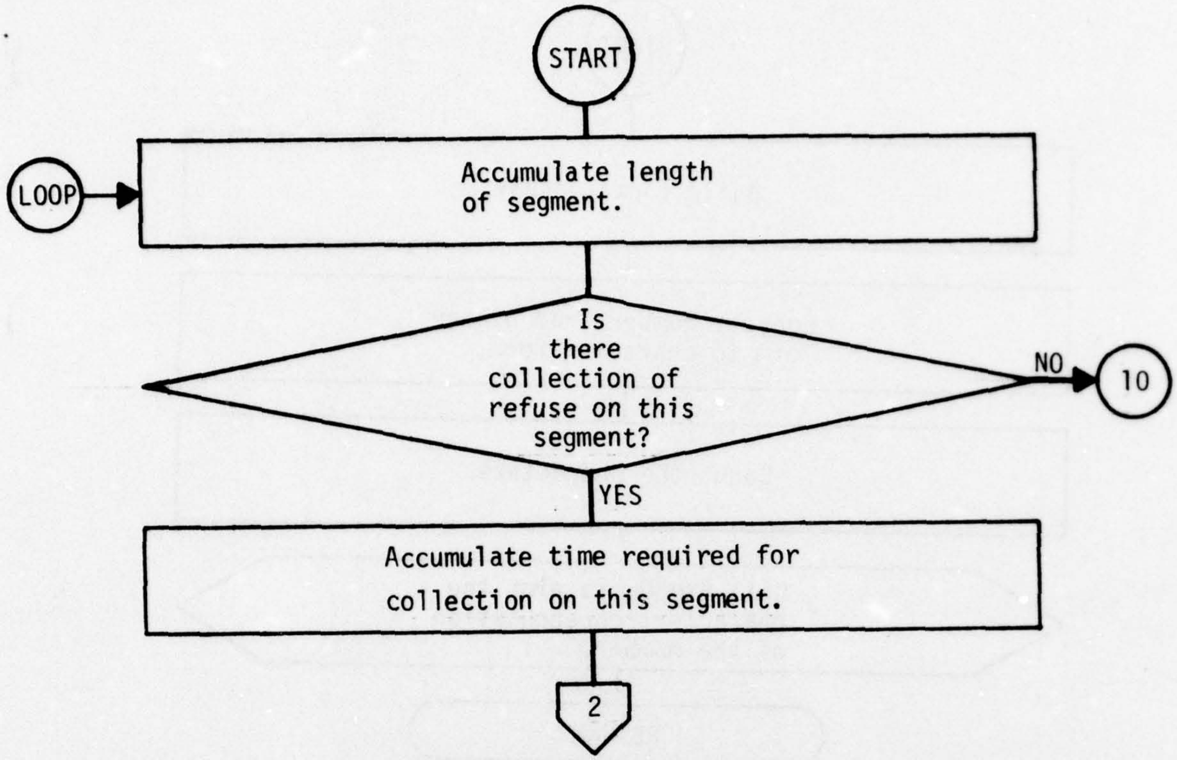
Subroutine STRIN


START
Convert whole and fractional hours to hours and minutes, rounding to the nearest minute.
Convert hours and minutes to character form using an ENCODE statement.
RETURN
END

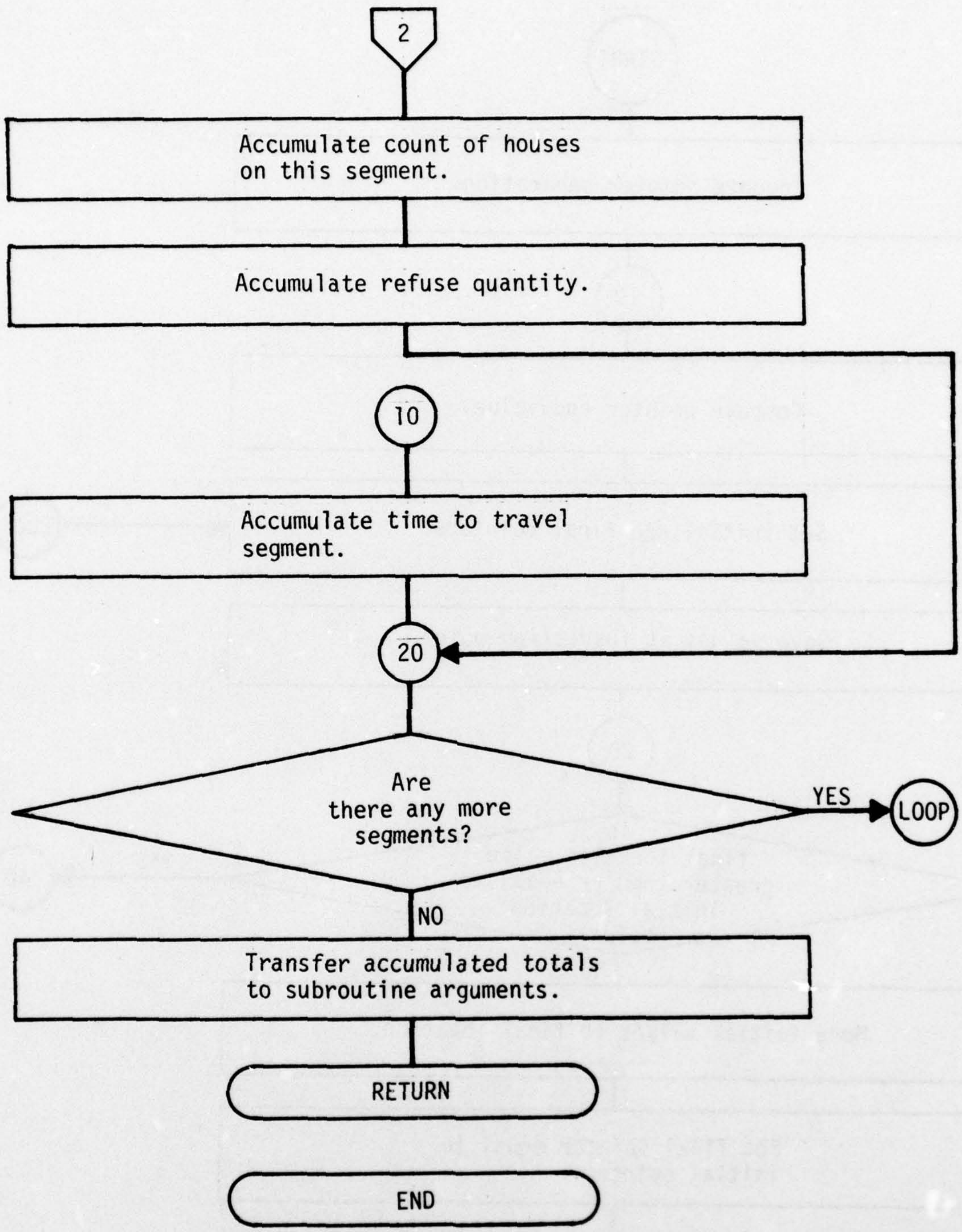
Function HM



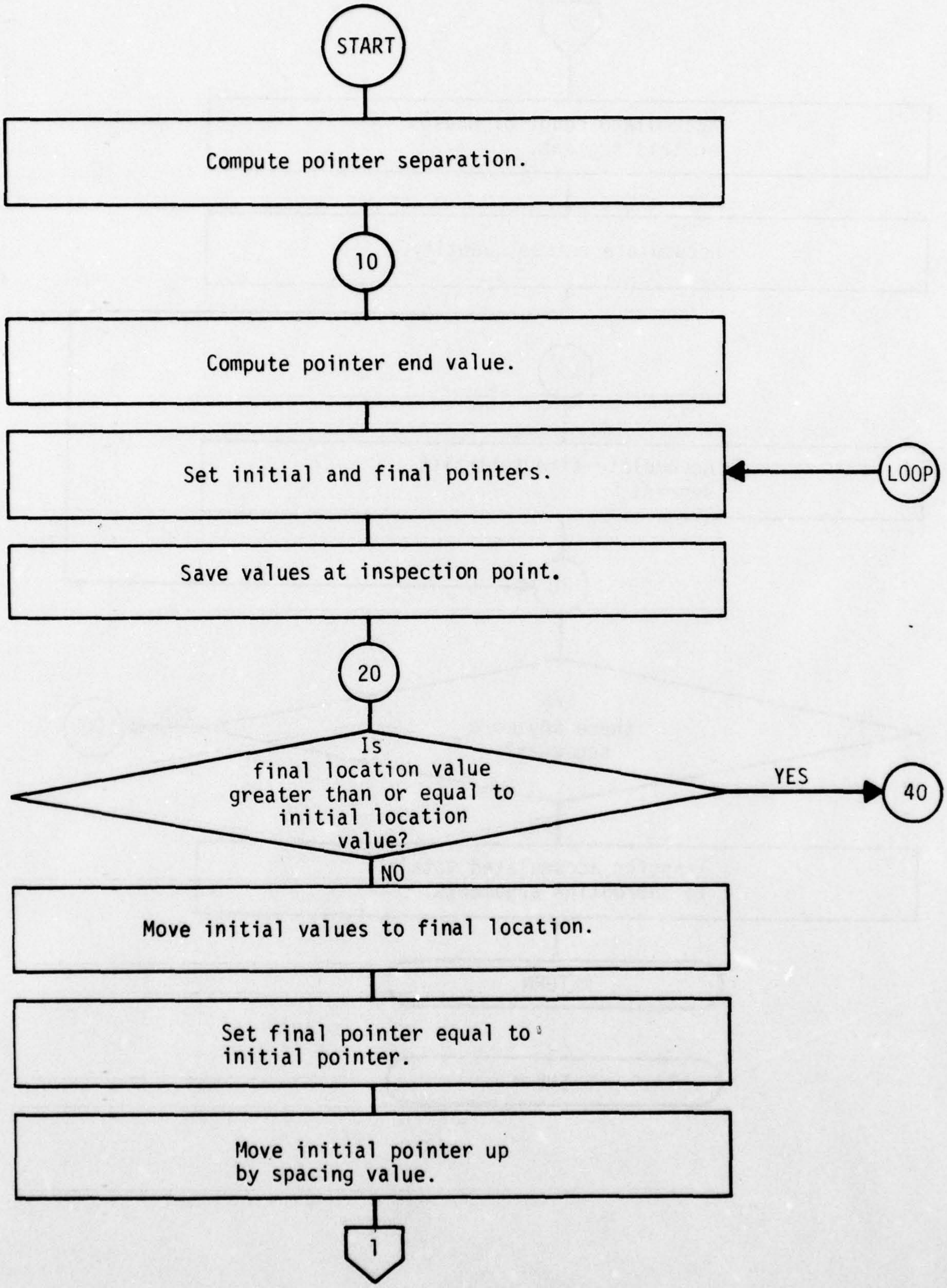
Subroutine NUMBER



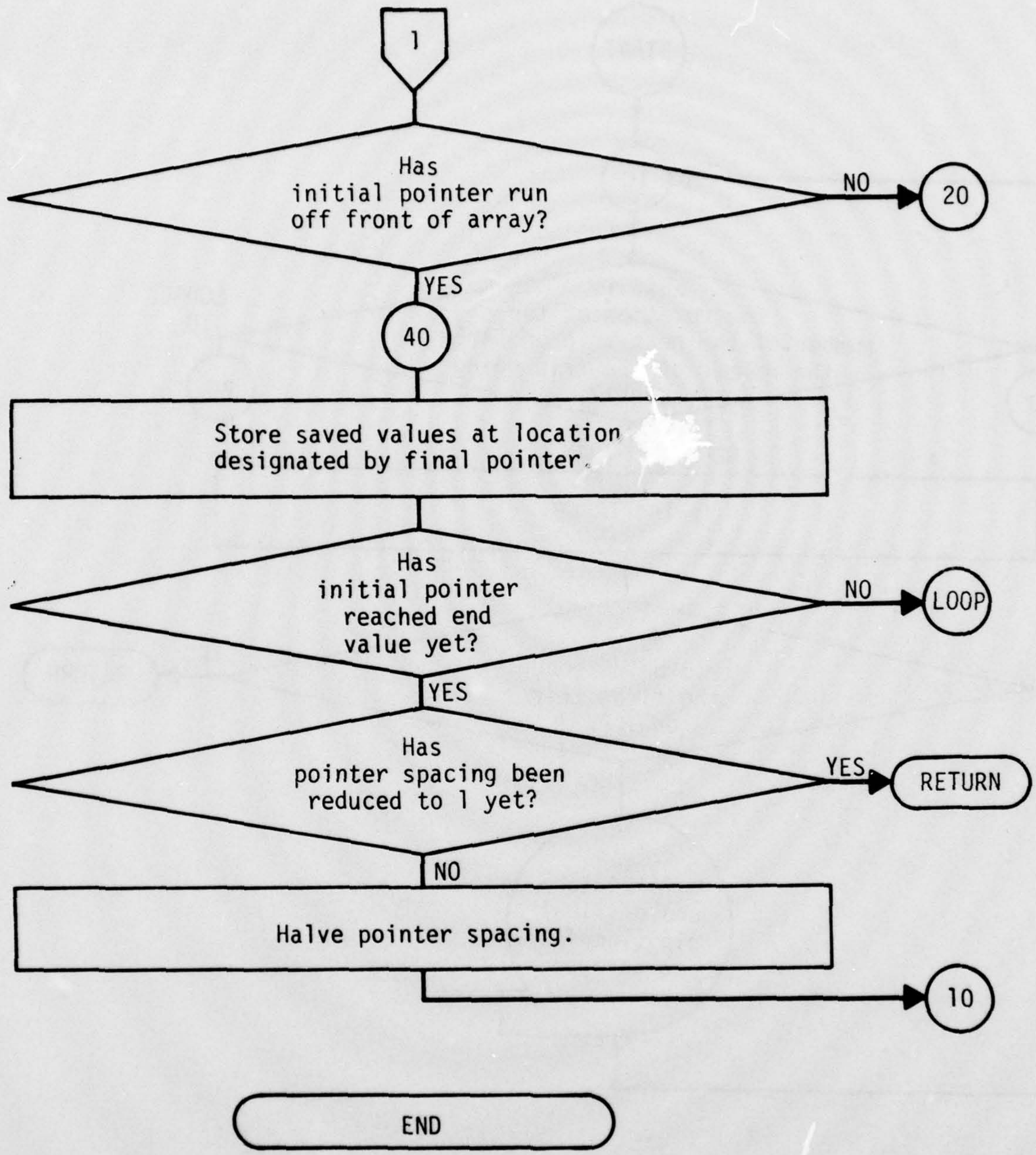
Subroutine CUMTD



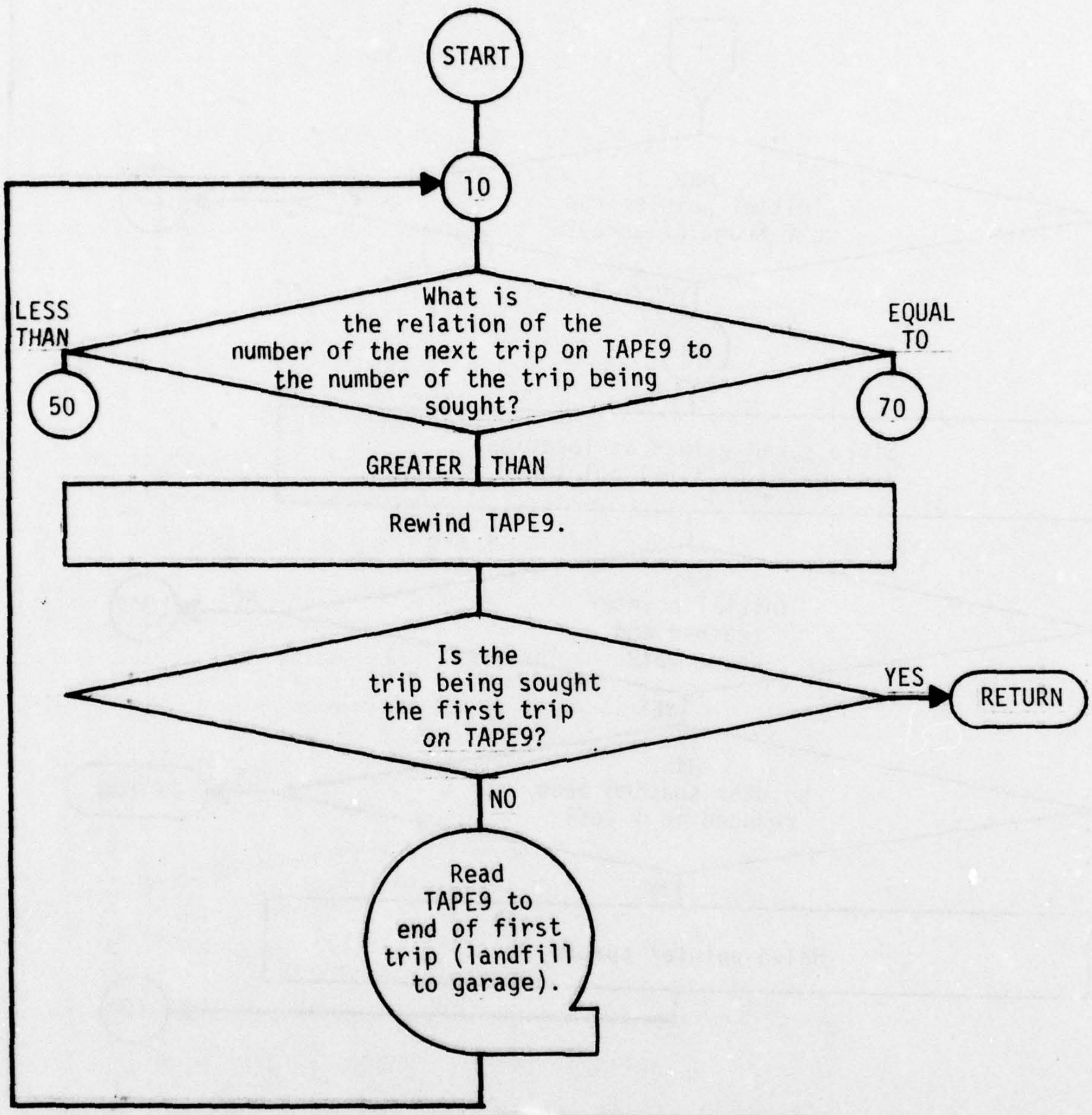
Subroutine CUMTD



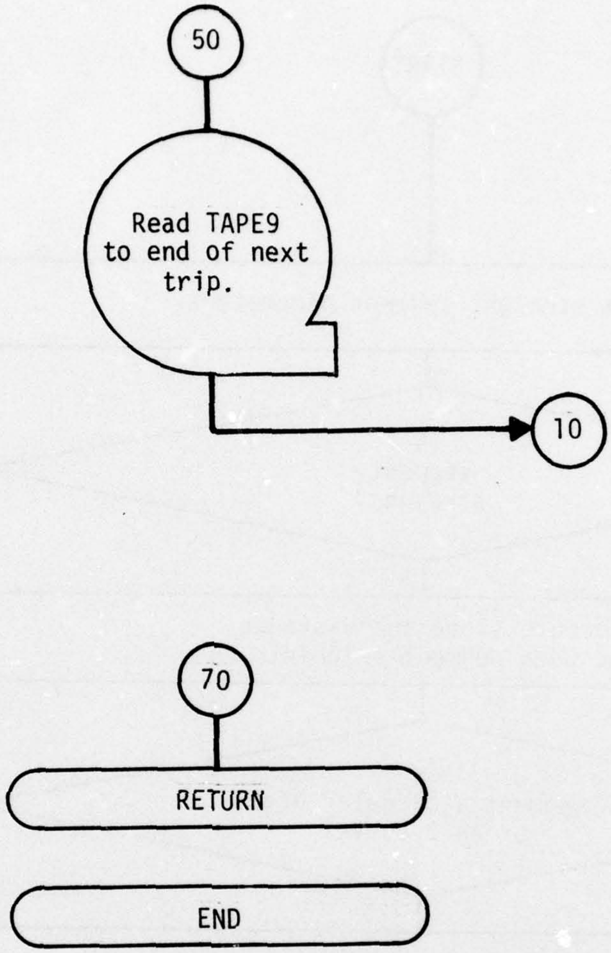
Subroutine SHLSRT



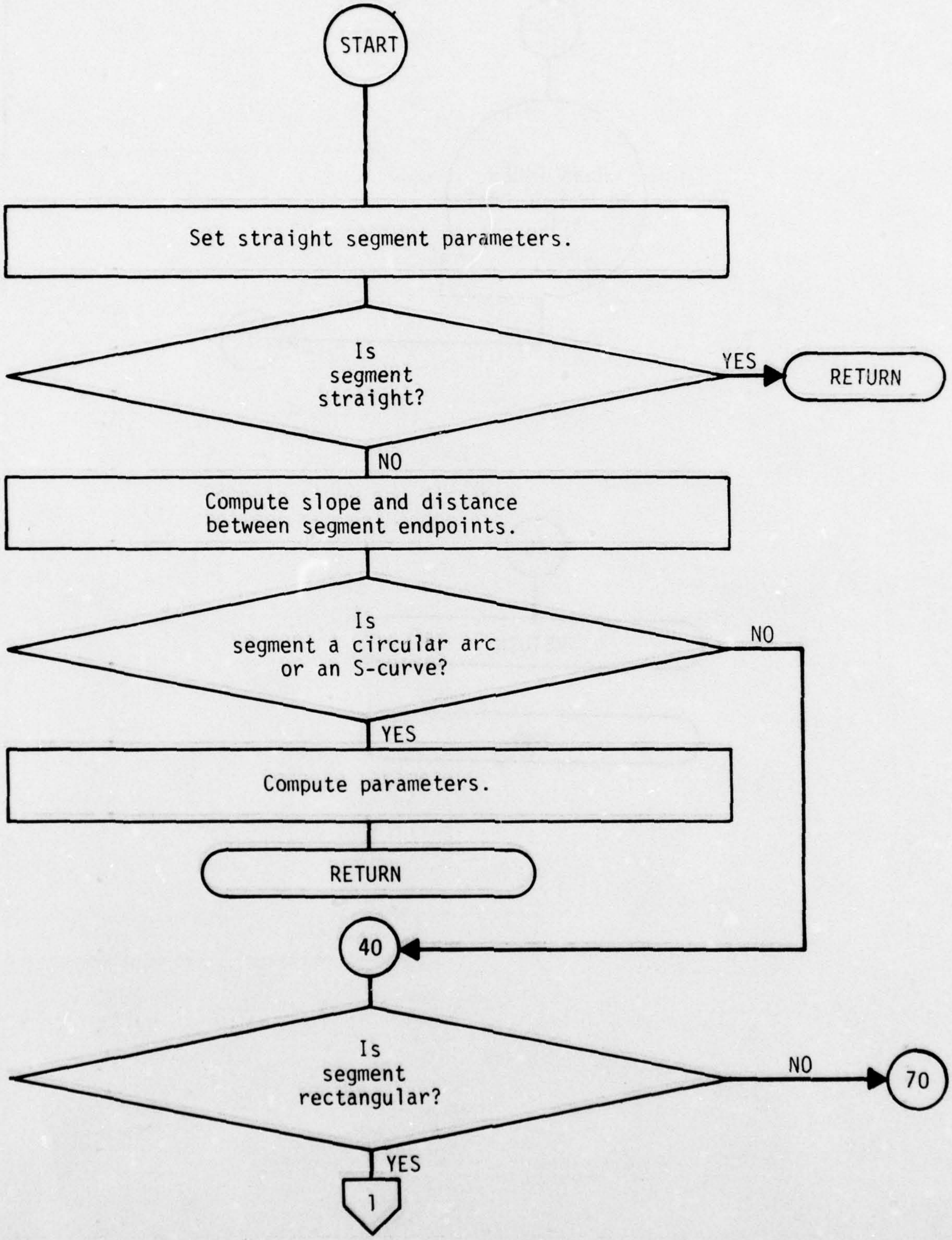
Subroutine SHLSRT



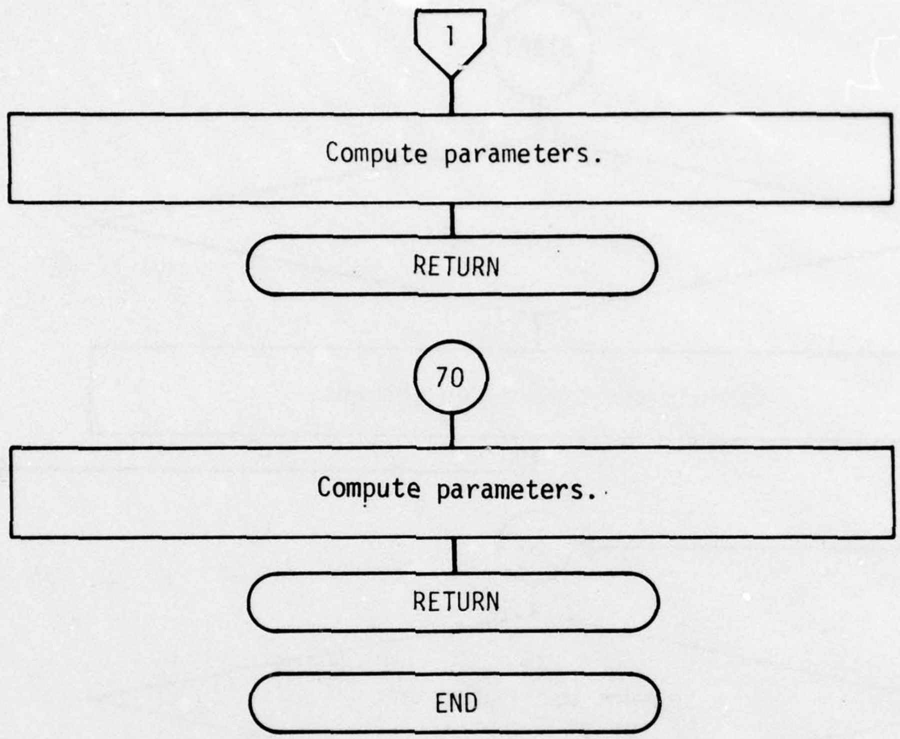
Subroutine POSIT9



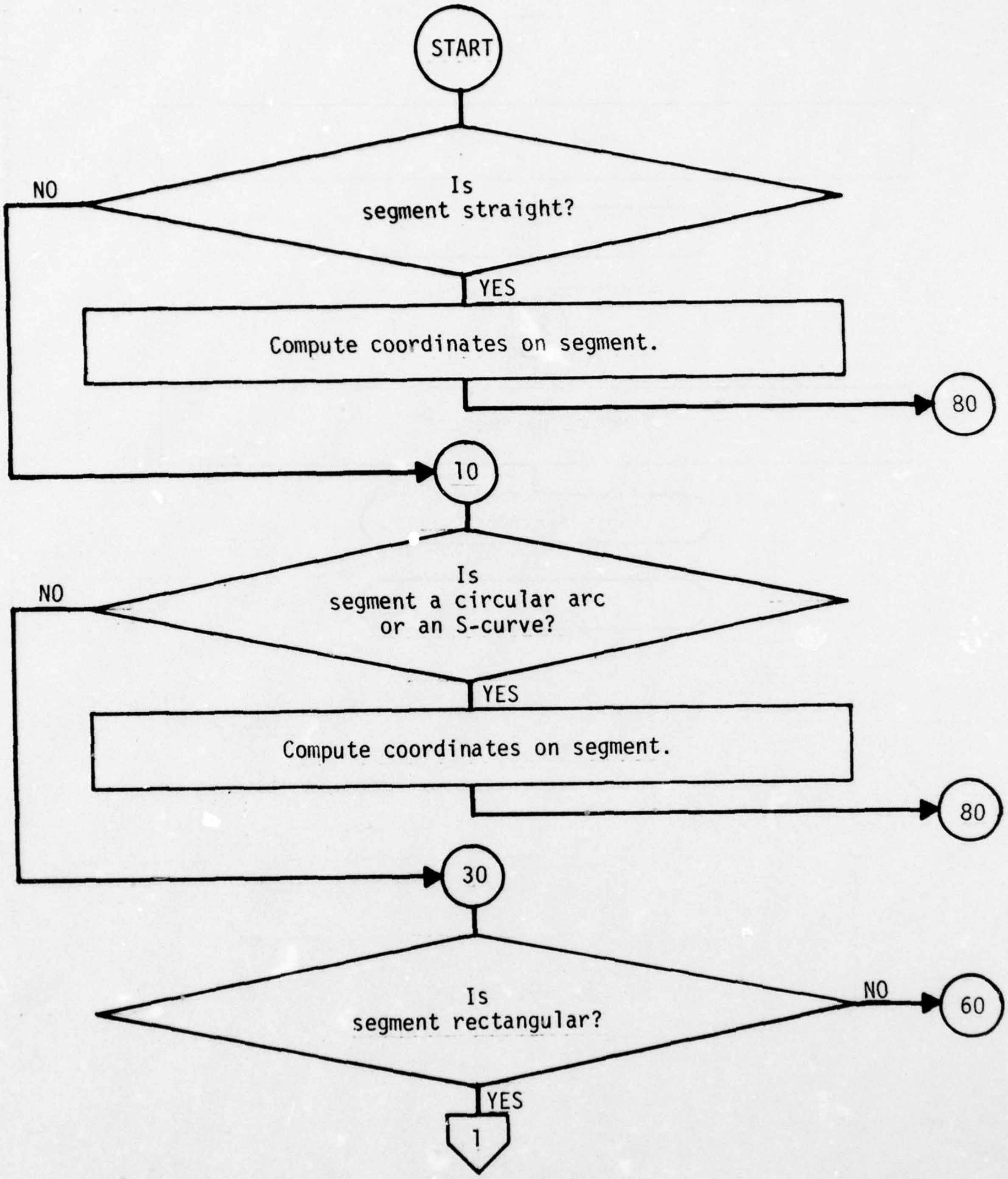
Subroutine POSIT9



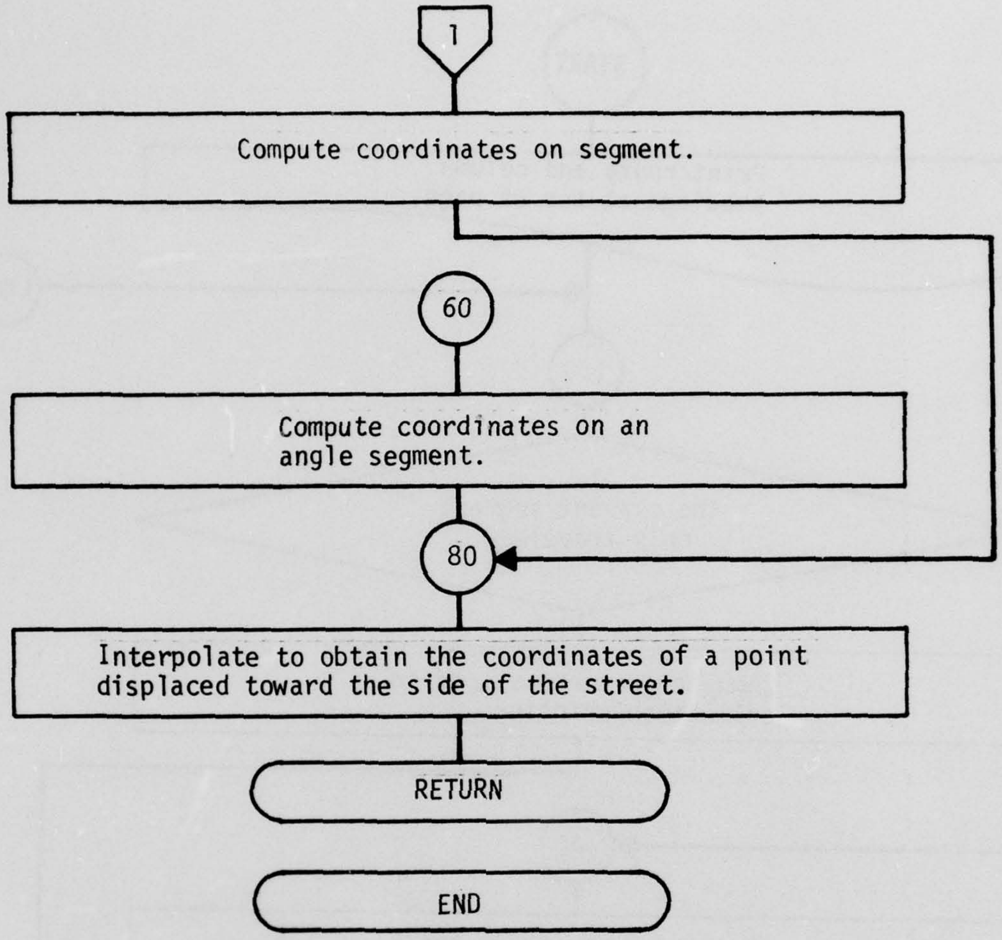
Subroutine SHAPCOM



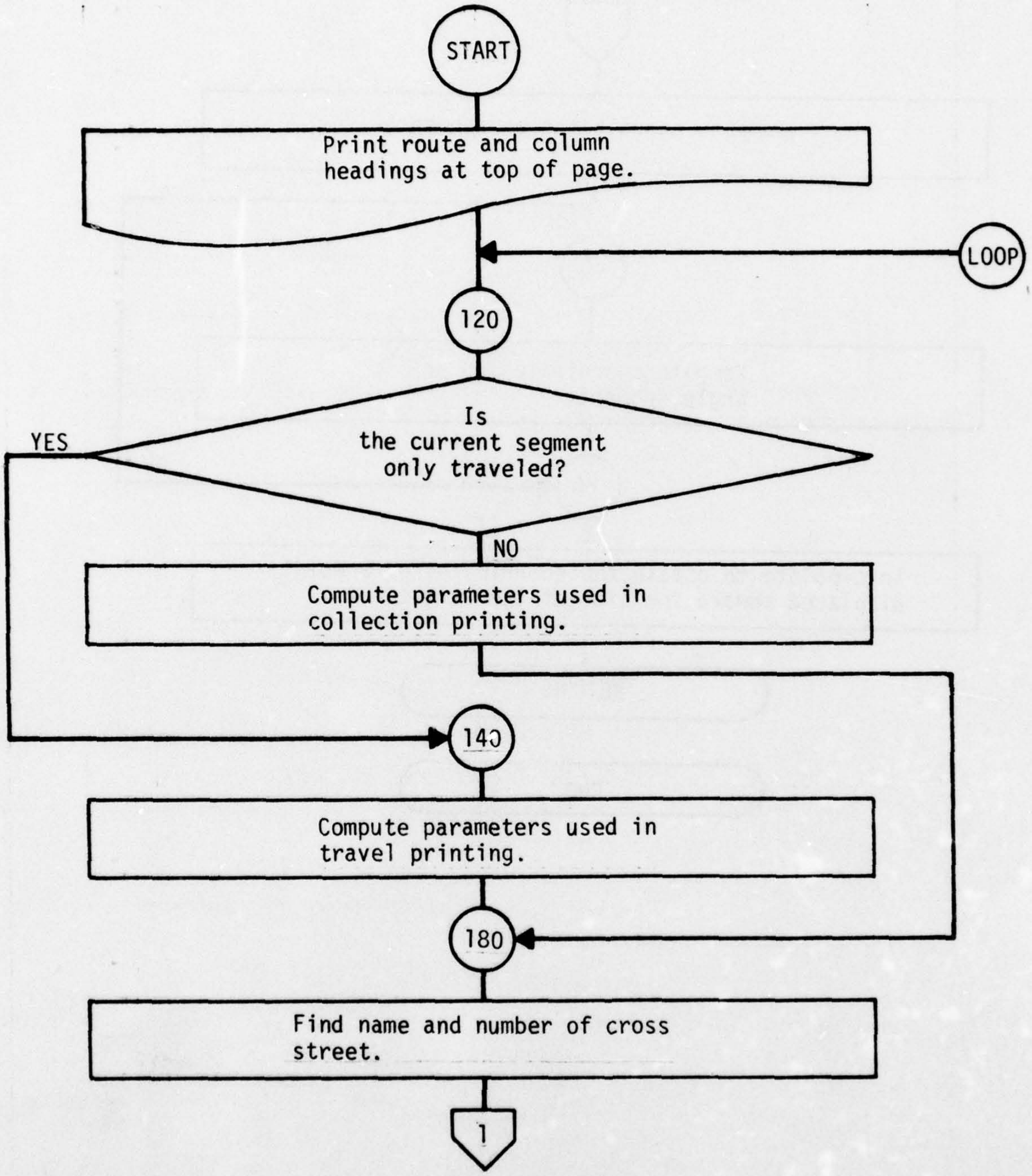
Subroutine SHAPCOM



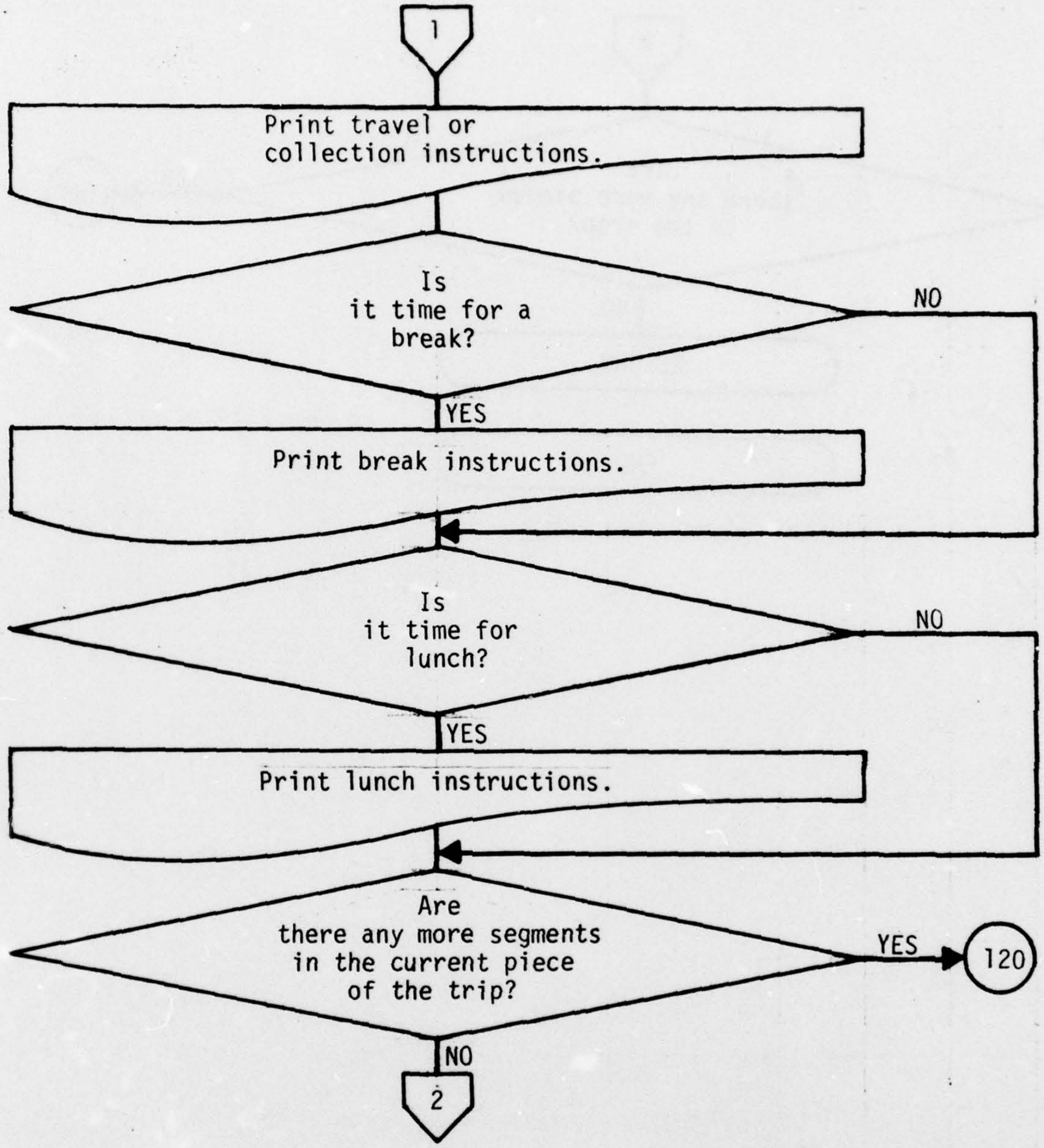
Subroutine COORD



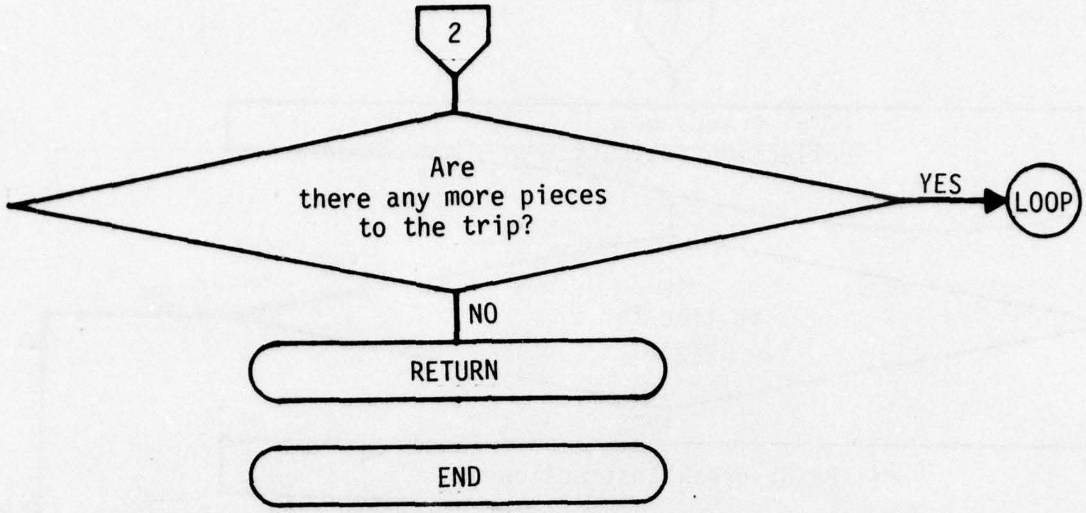
Subroutine COORD



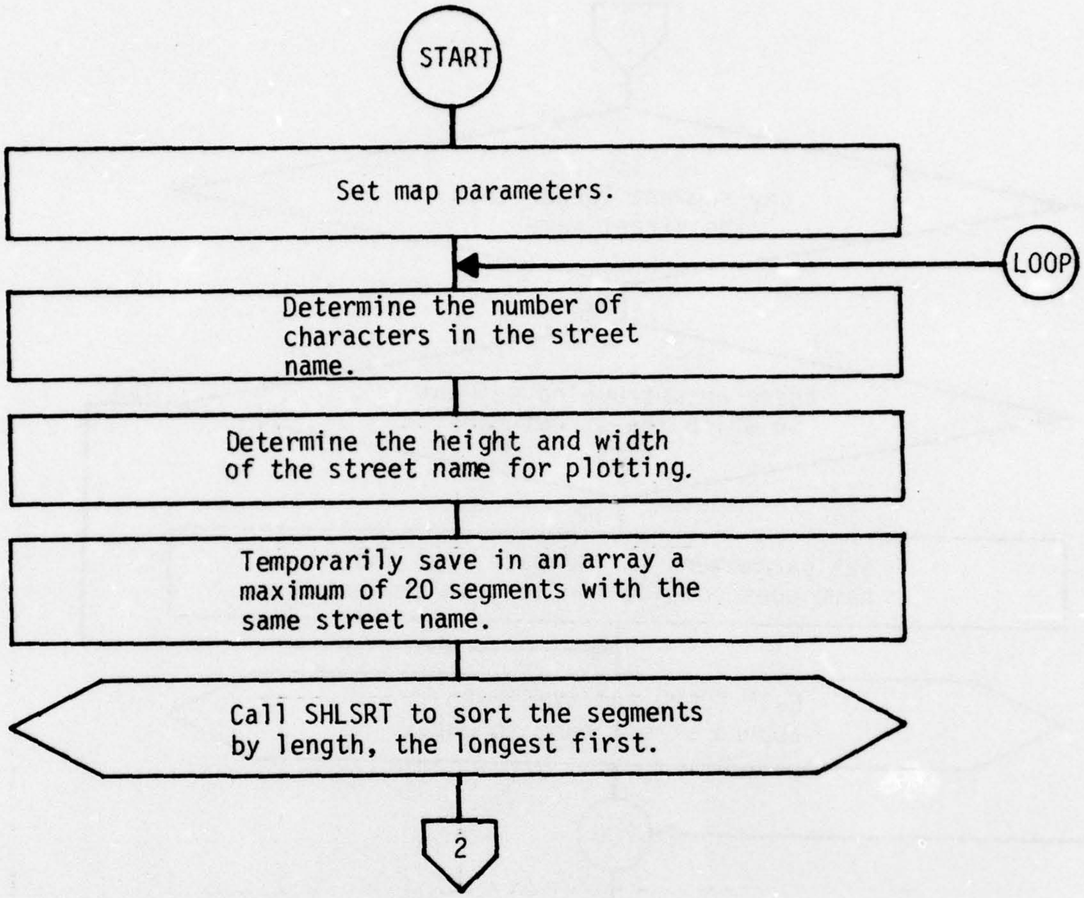
Subroutine PRSCHED



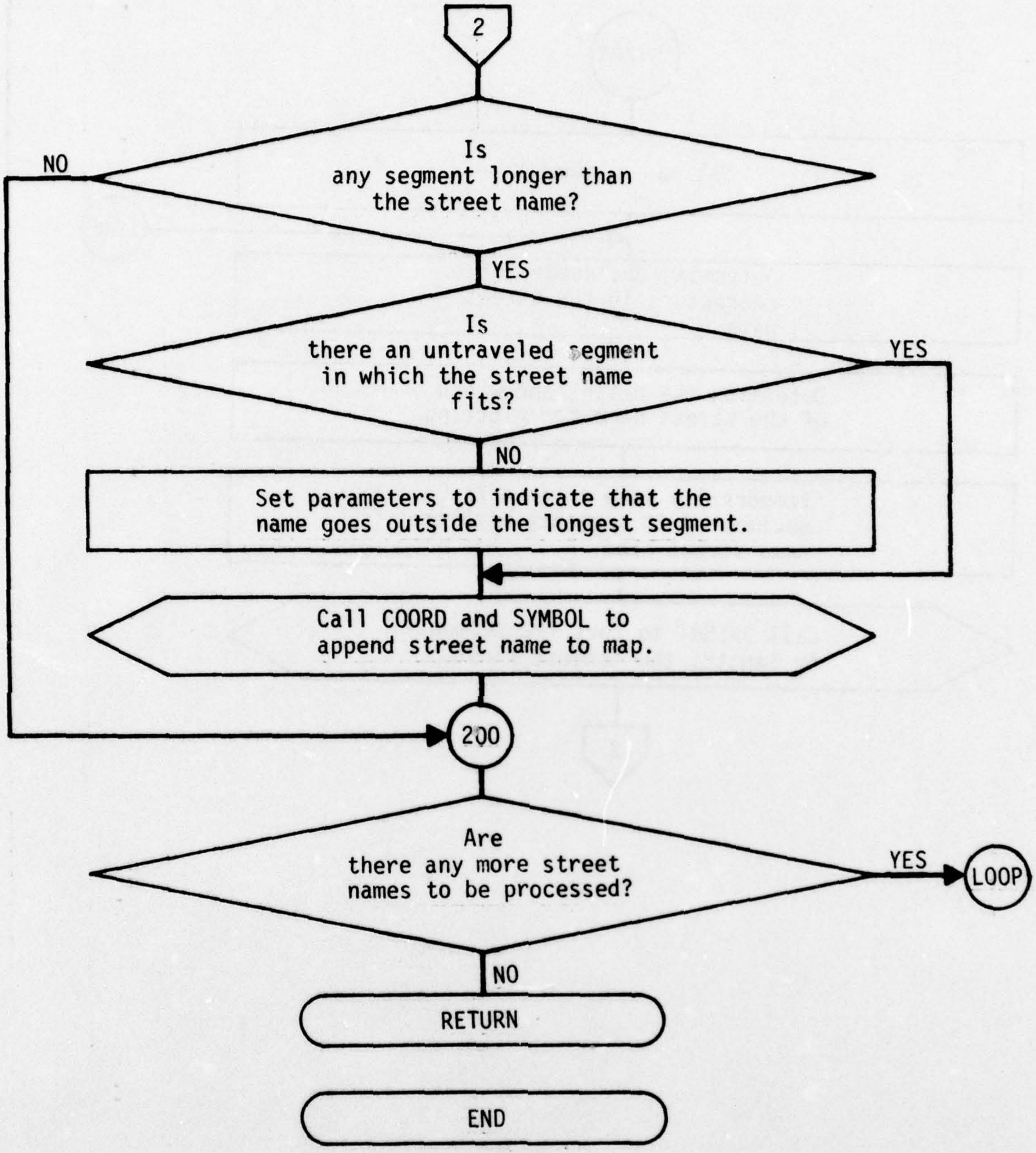
Subroutine PRSCHED



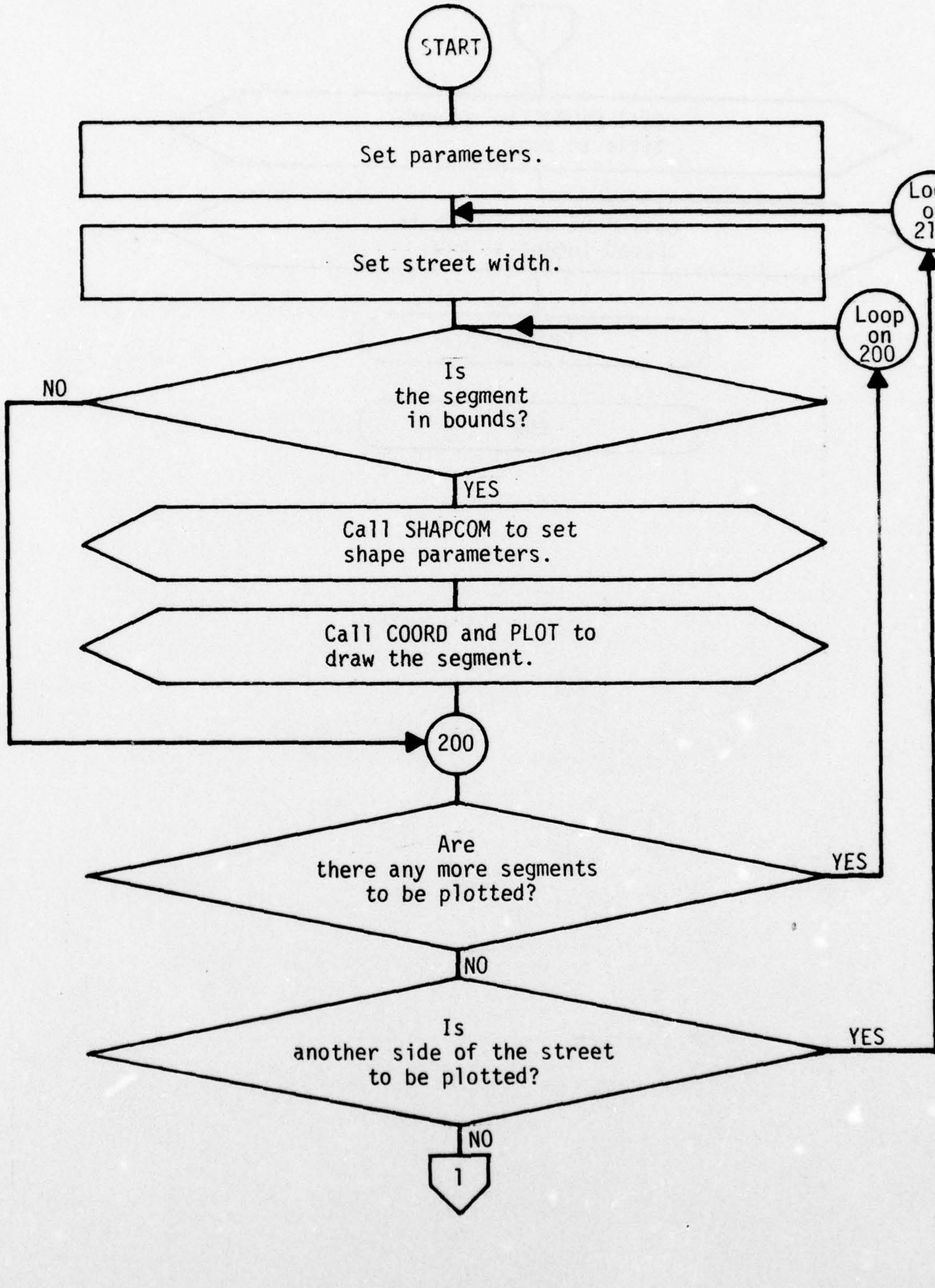
Subroutine PRSCHED



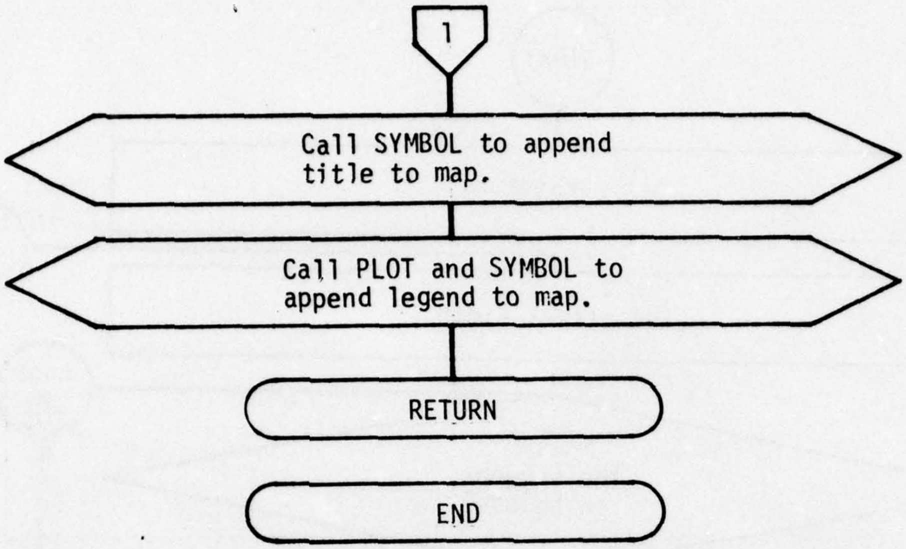
Subroutine STNAME



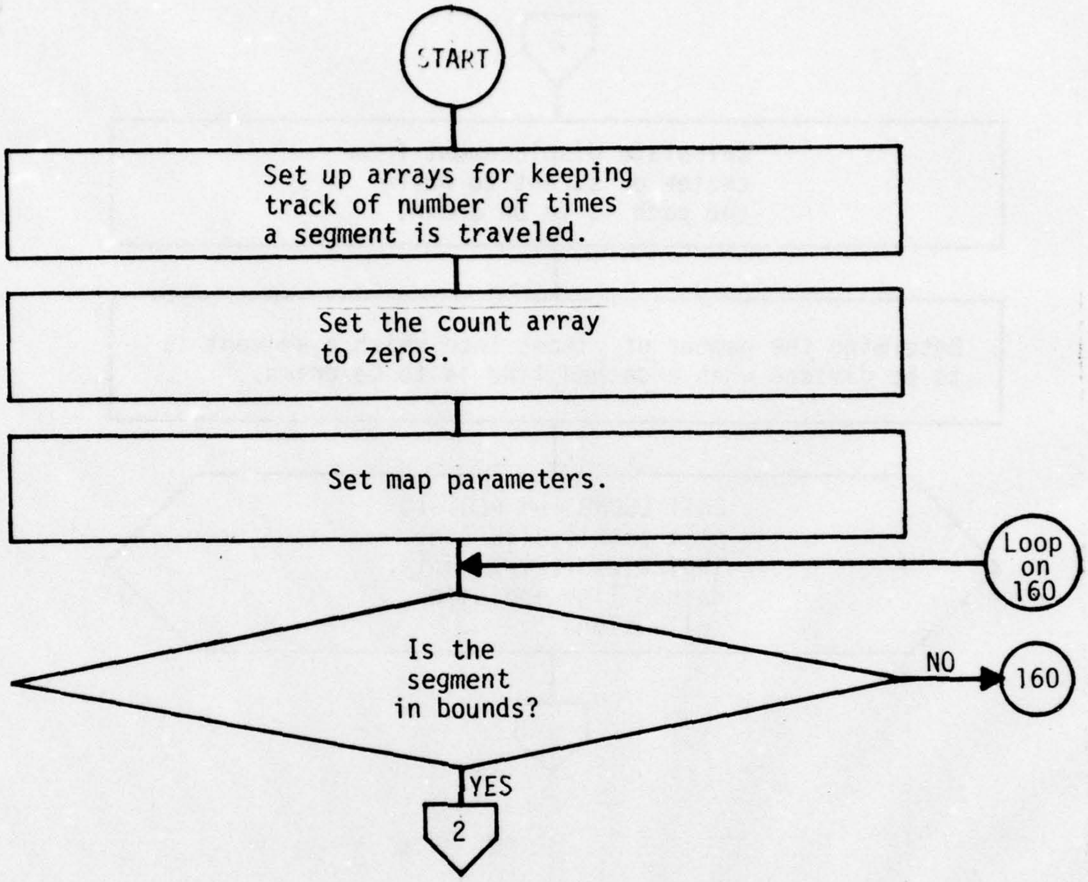
Subroutine STNAME



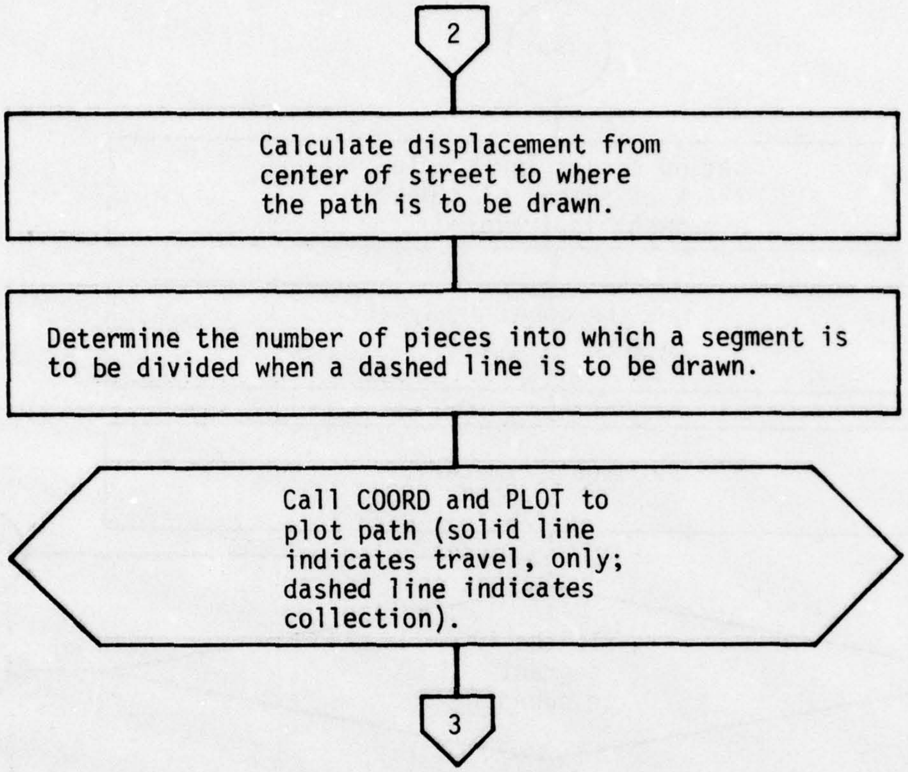
Subroutine MAPPLT



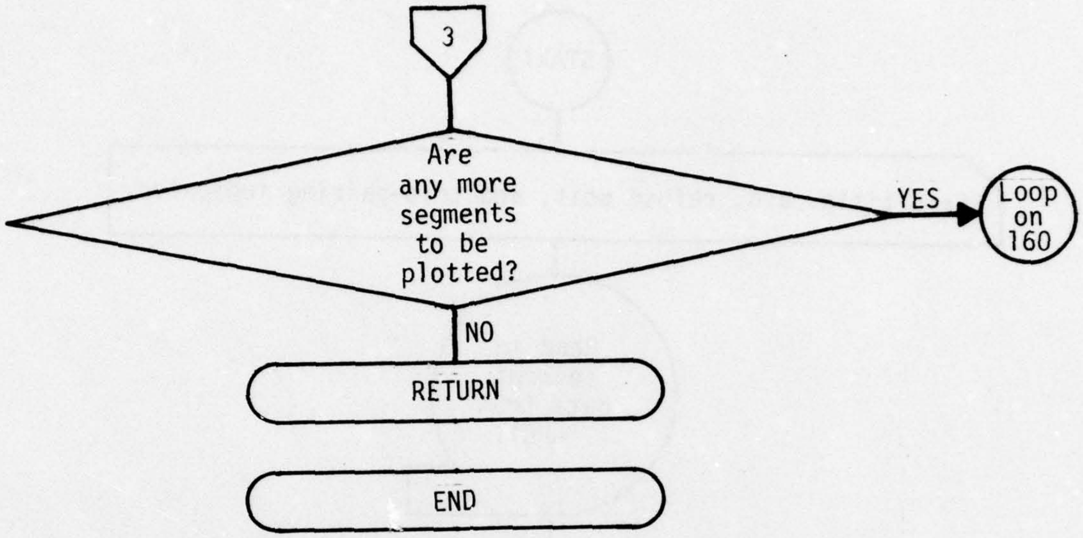
Subroutine MAPPLT



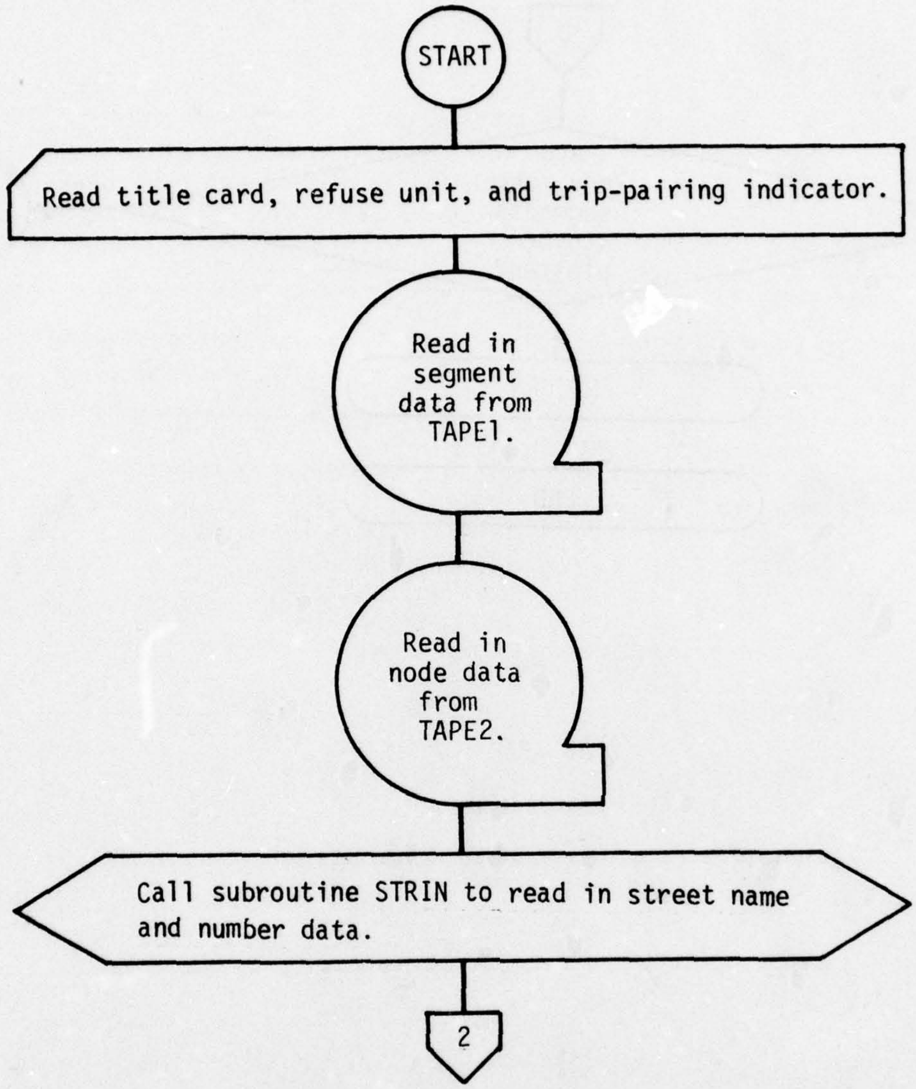
Subroutine PATHPLT



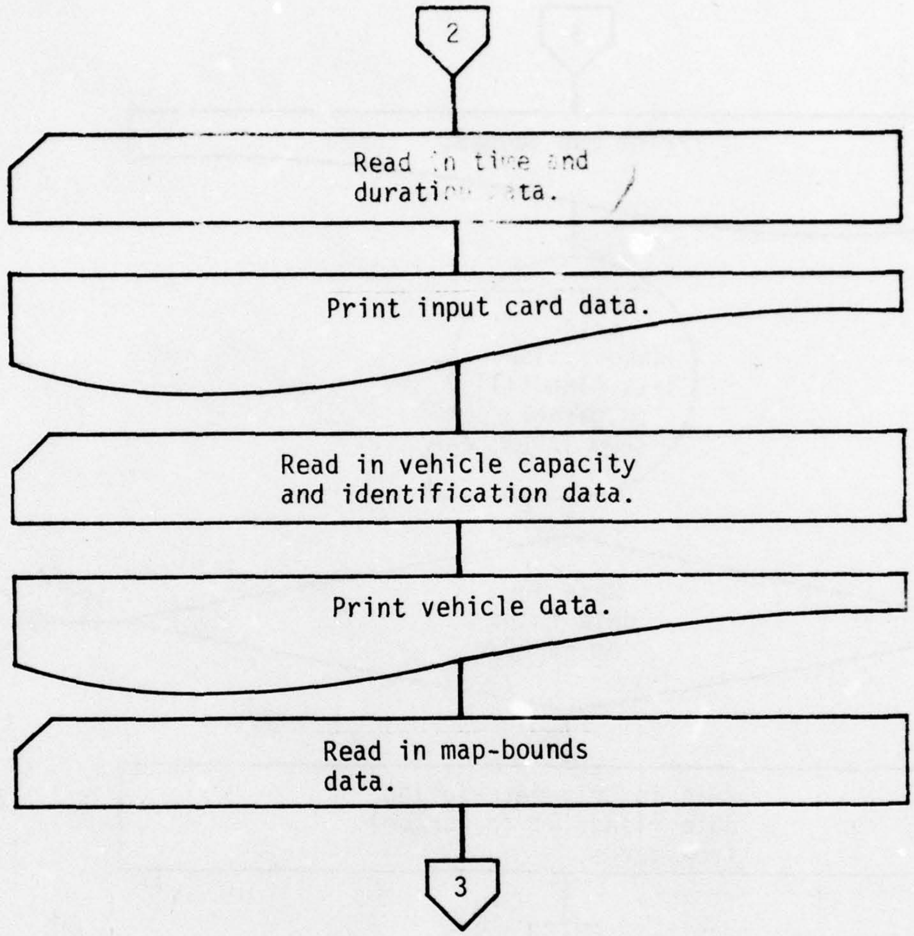
Subroutine PATHPLT



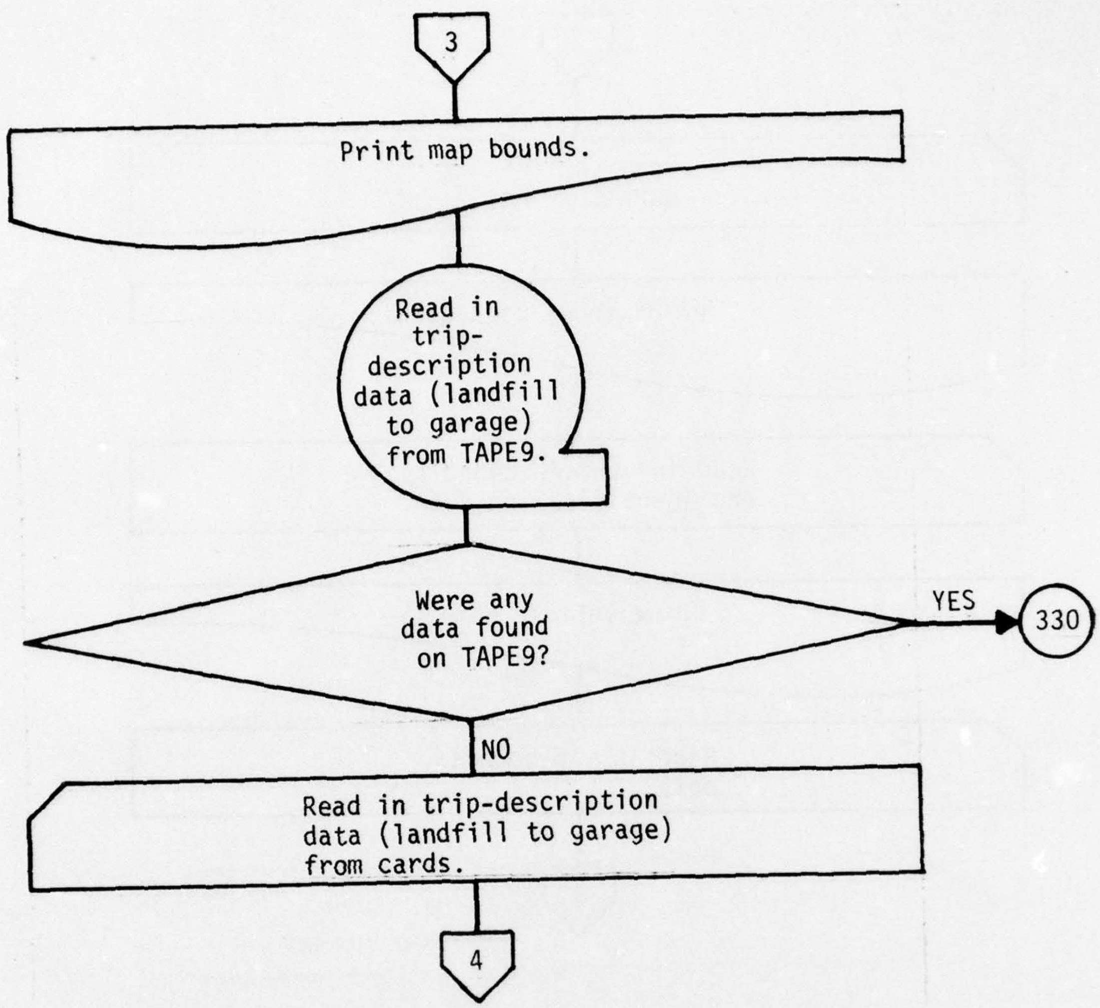
Subroutine PATHPLT



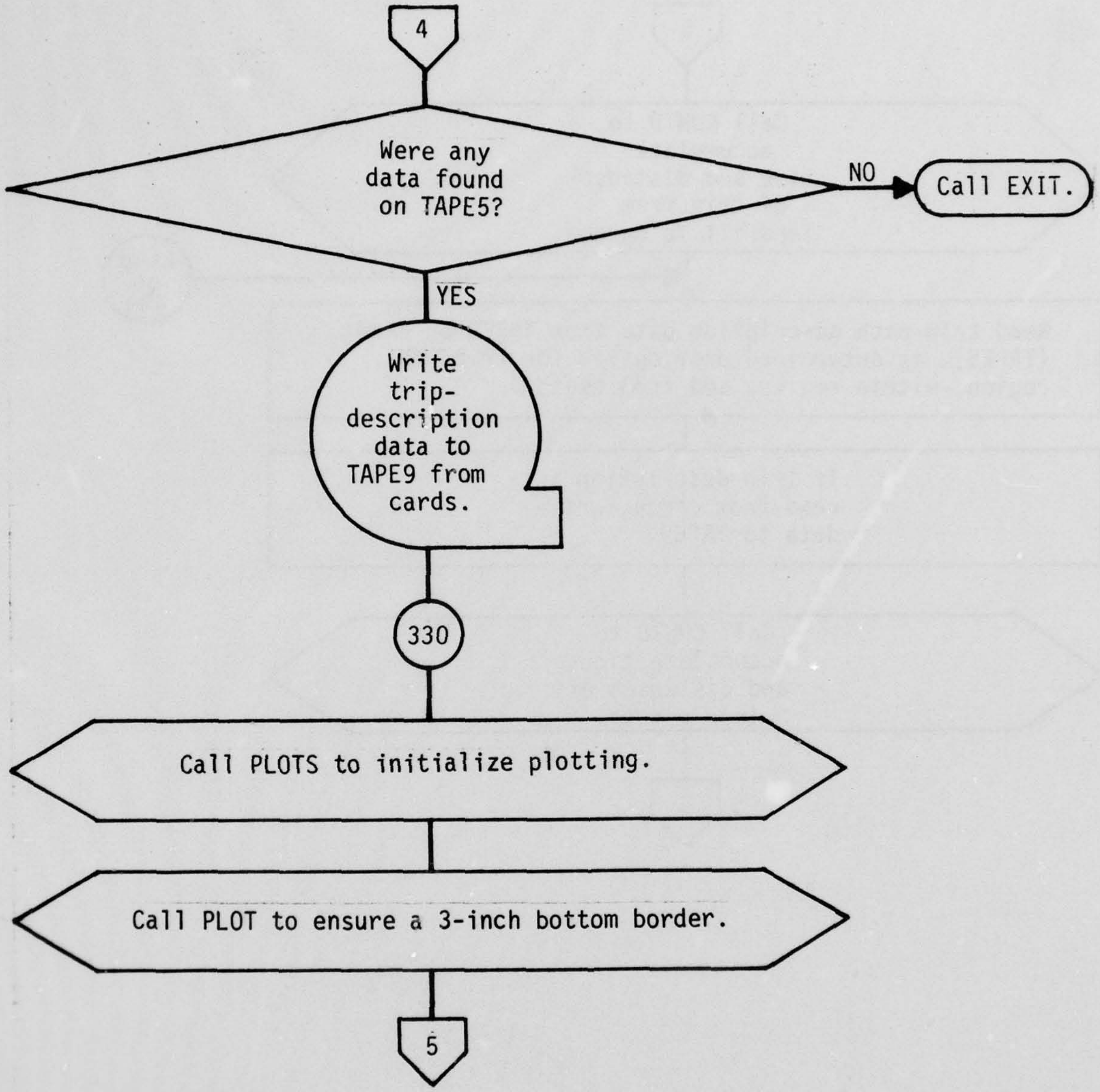
Program PHASE4

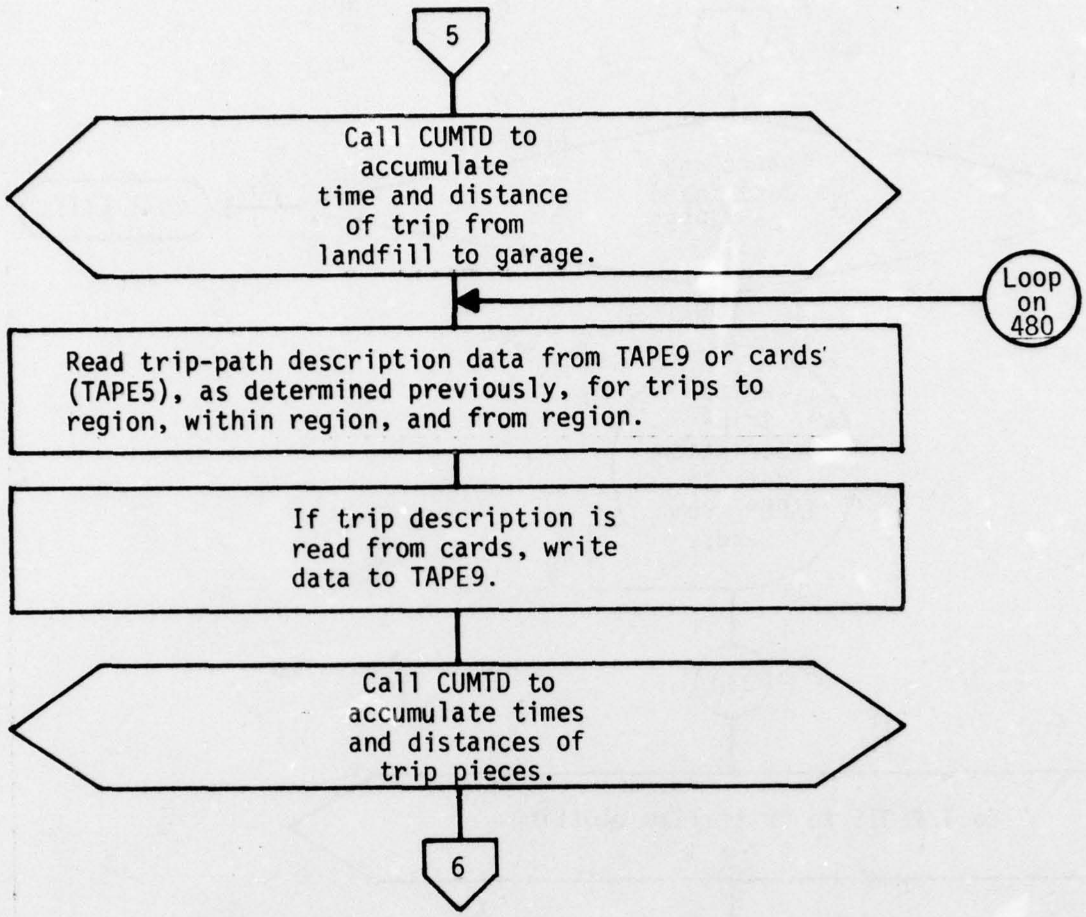


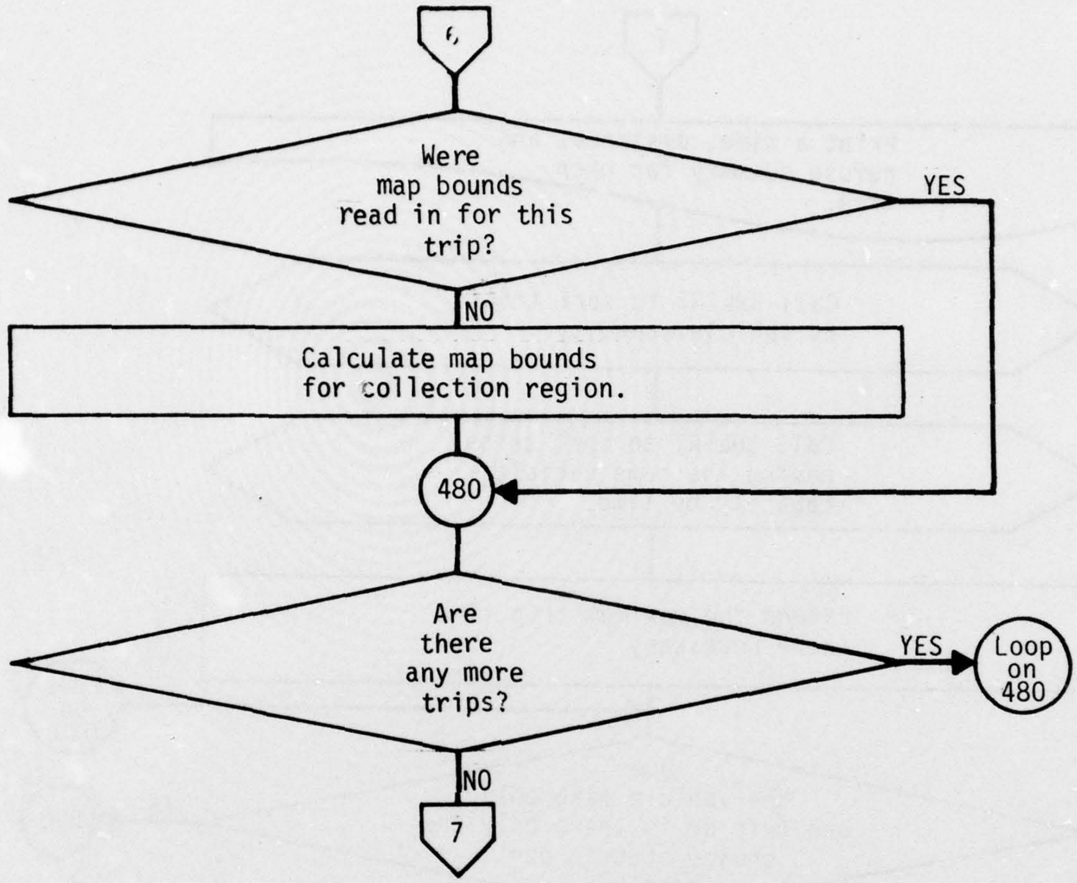
Program PHASE4



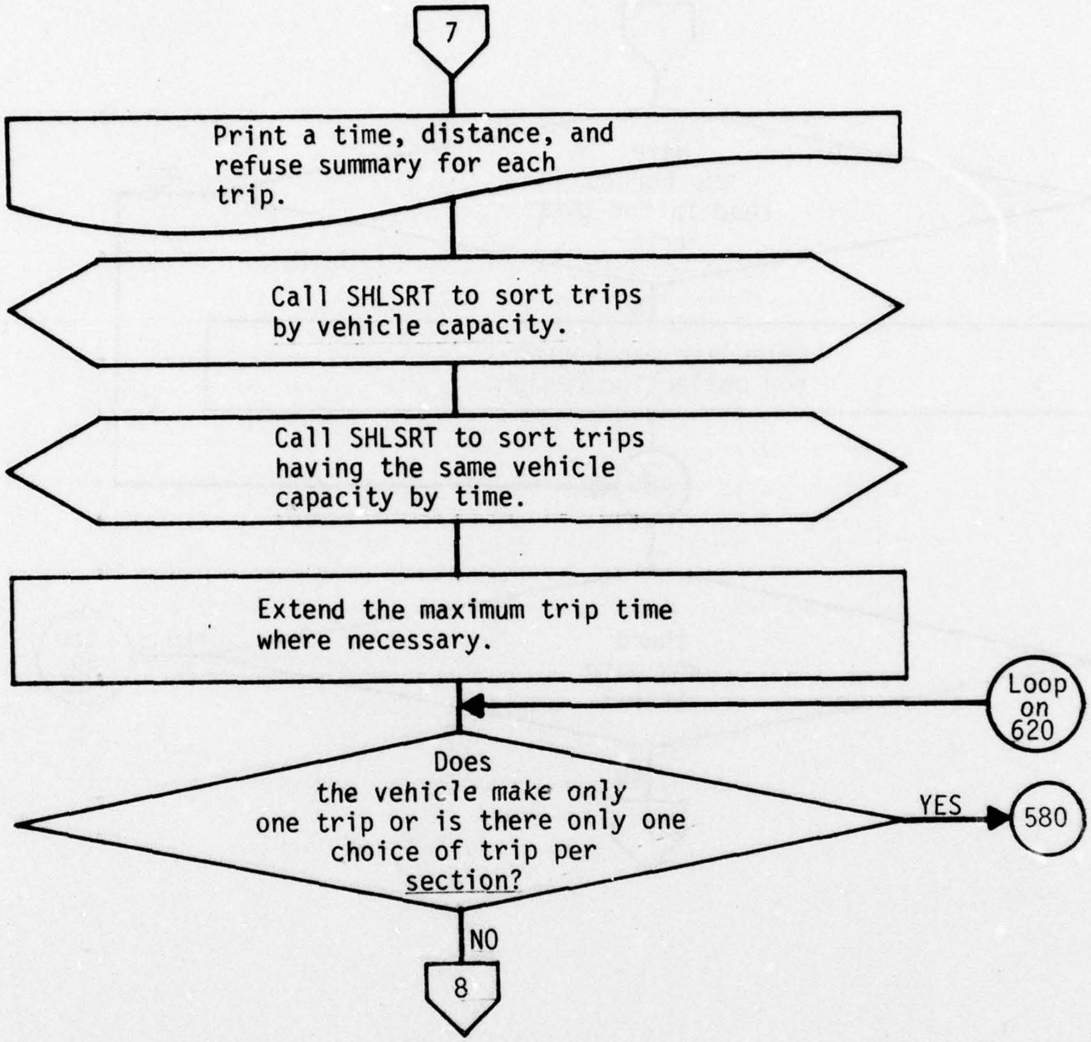
Program PHASE4


4
Were any data found on TAPE5?
NO
Call EXIT.
YES
Write trip-description data to TAPE9 from cards.
330
Call PLOTS to initialize plotting.
Call PLOT to ensure a 3-inch bottom border.
5

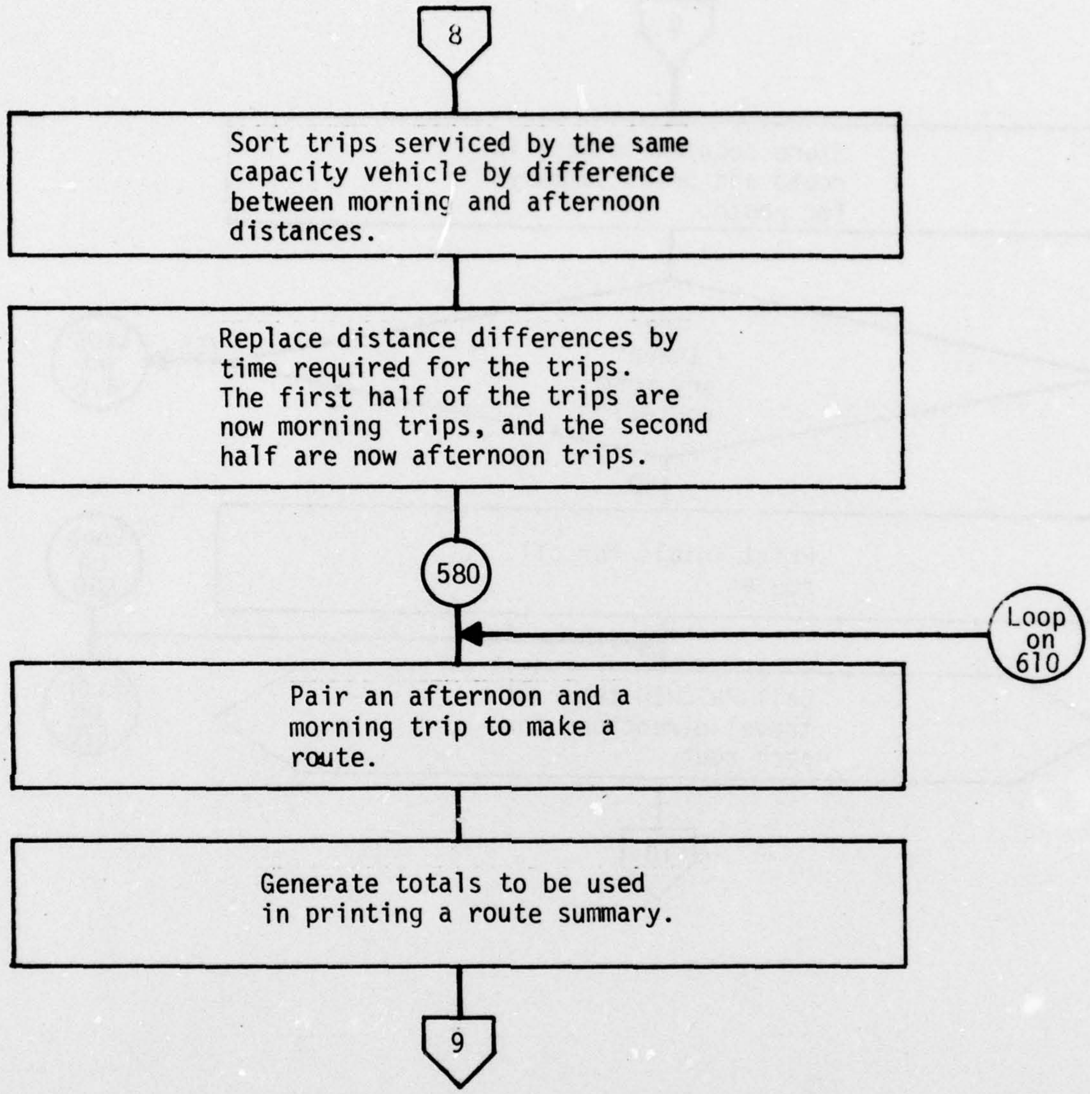

5
Call CUMTD to accumulate time and distance of trip from landfill to garage.
Loop on 480
Read trip-path description data from TAPE9 or cards (TAPE5), as determined previously, for trips to region, within region, and from region.
If trip description is read from cards, write data to TAPE9.
Call CUMTD to accumulate times and distances of trip pieces.
6



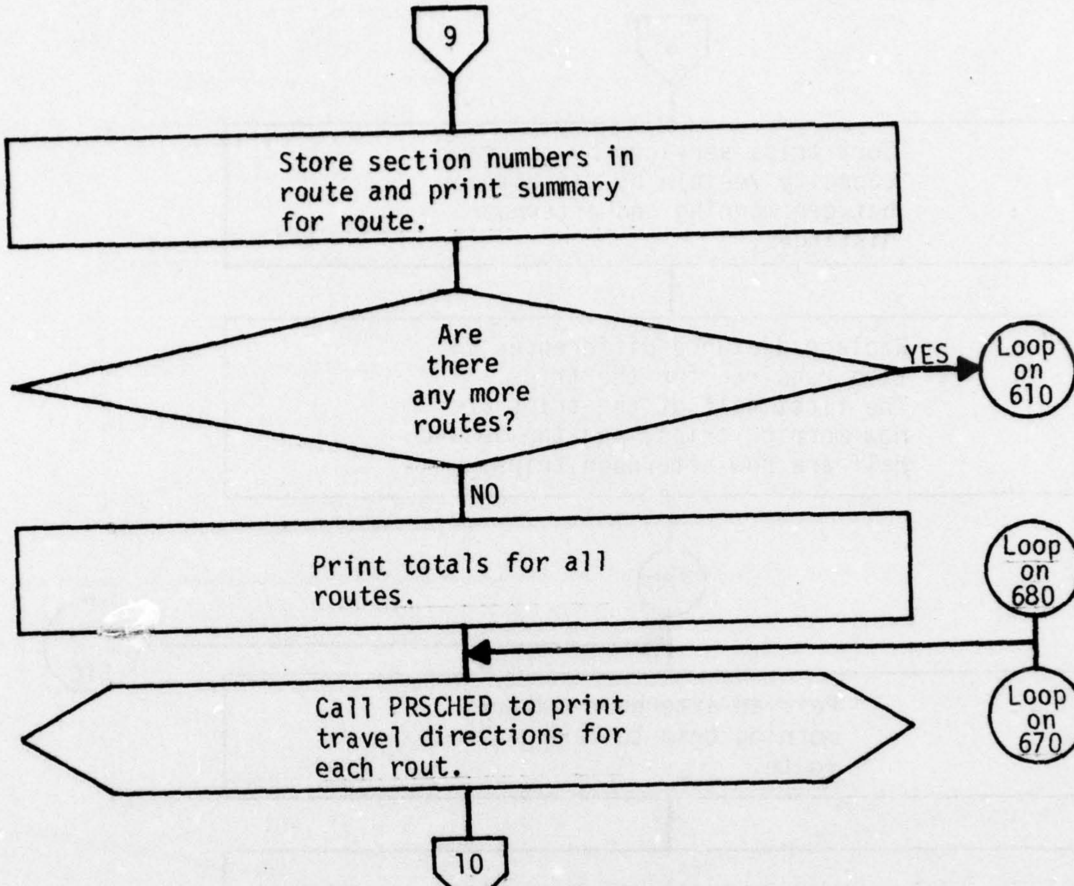
Program PHASE4



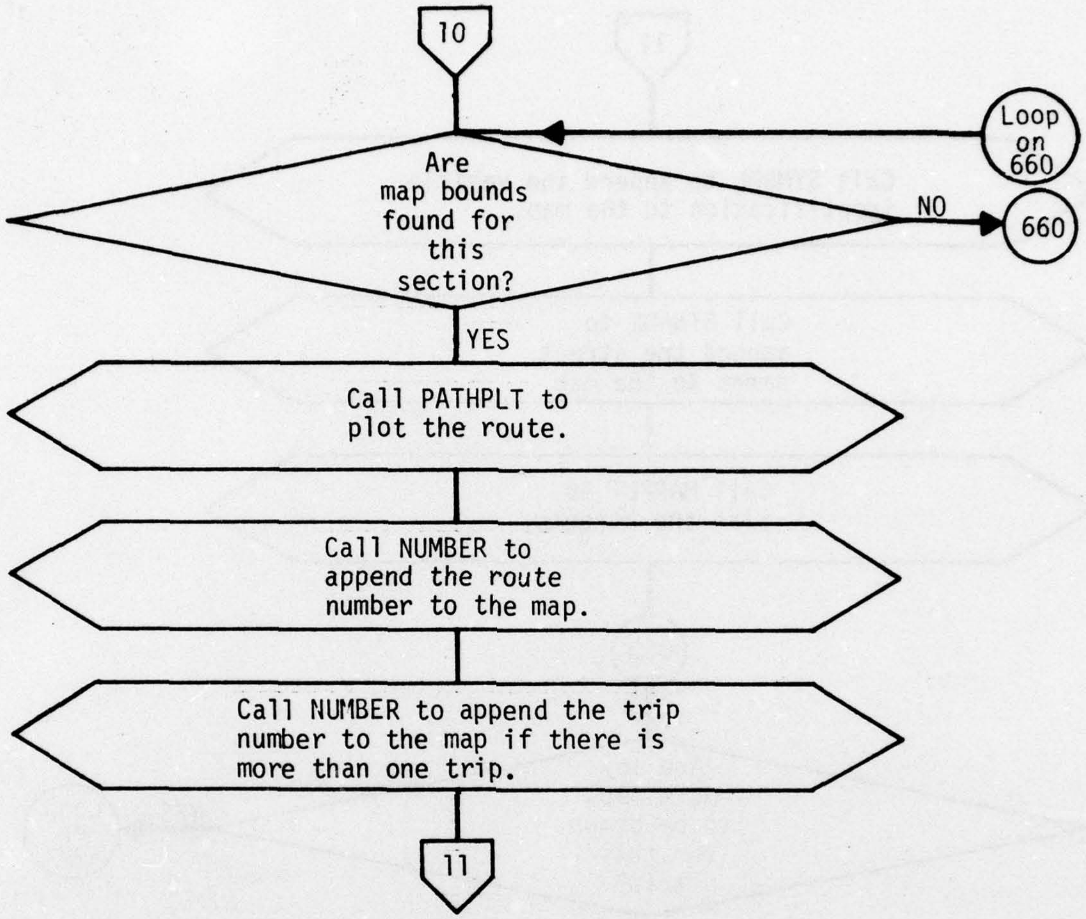
Program PHASE4



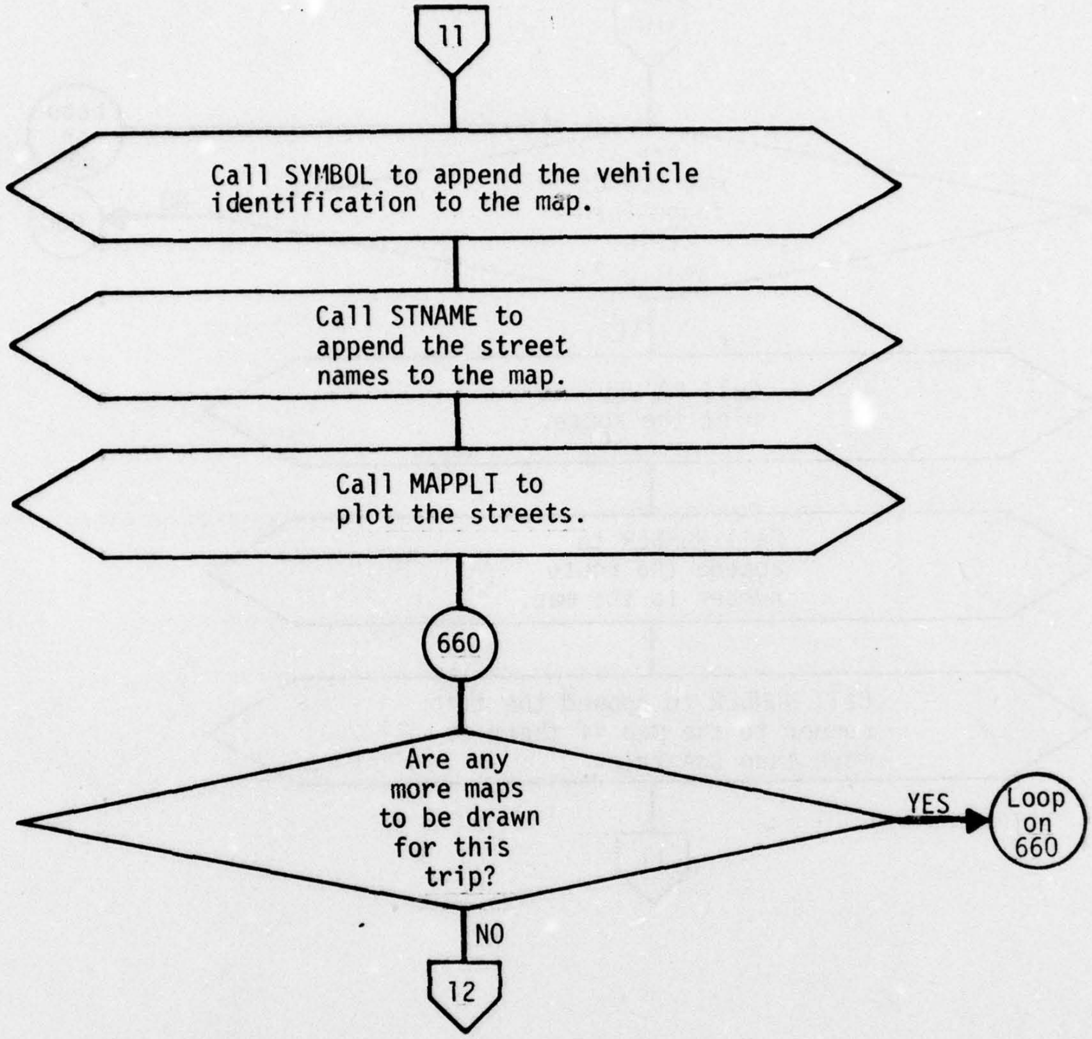
Program PHASE4



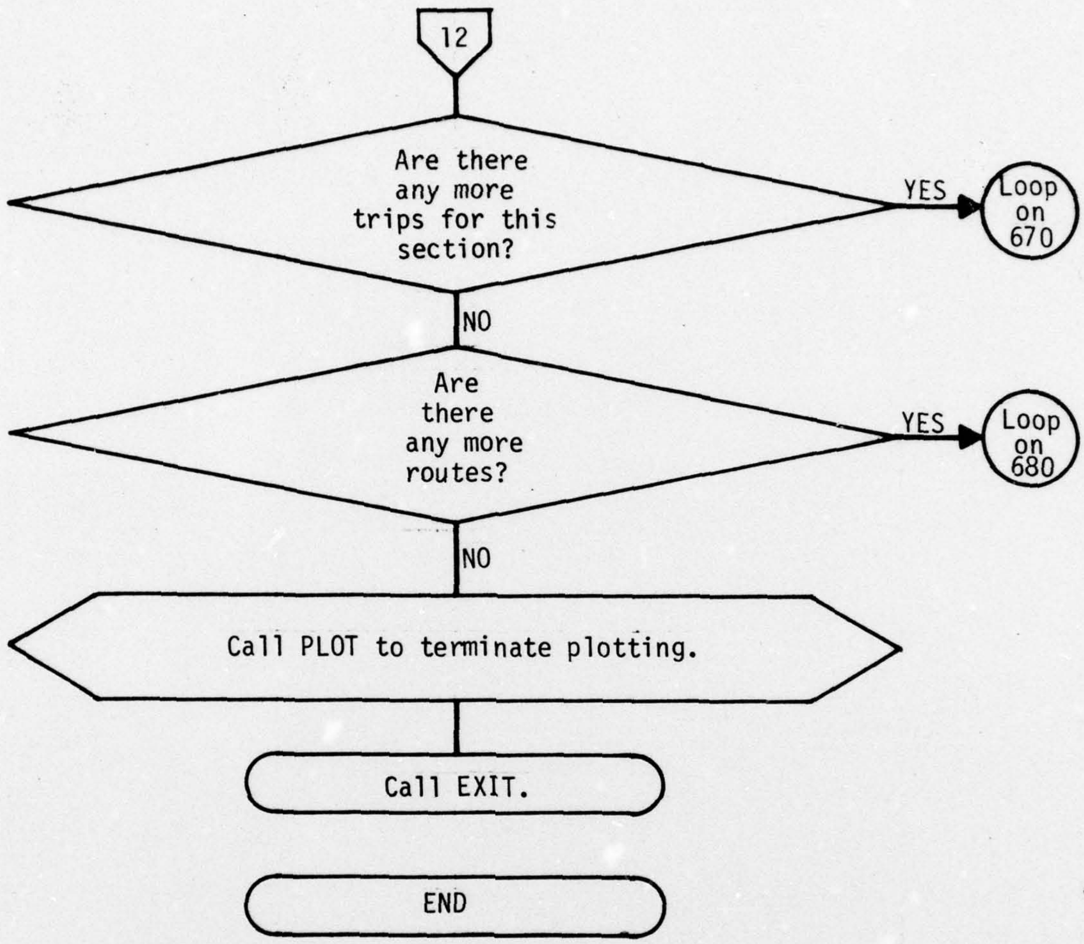
Program PHASE4



Program PHASE4



Program PHASE4



Program PHASE4

(The reverse of this page is blank.)

APPENDIX B
PROGRAM LISTINGS

```
      SUBROUTINE STRIN

C        LATEST CHANGES --
C        JAN 5. 1977.  HJI.  ORIGINAL VERSION.

      COMMON /STREETS/ NSTRS.NUMSTR(300),NAMSTR(7,300),NONAME(7),
     1  NMFILL(7),NMGAR(7)
      COMMON /TEMSTG/ IBUFF(800)

C        FIRST LOOK FOR STREET DATA ON TAPE 3.
      NSTRS=0
   10 BUFFER IN (3,1) (IBUFF(1),IBUFF(800))
      IF (UNIT(3)) 20,20,20
   20 IF (LENGTH(3) .LE. 0) GO TO 50
      DO 30 I=1,100
      IJ=100+7*(I-1)
      NUMSTR(NSTRS+I)=IBUFF(I)
        DO 30 J=1,7
   30   NAMSTR(J,NSTRS+I)=IBUFF(J+IJ)
      DO 40 I=1,100
      IF (IBUFF(I) .EQ. 0) GO TO 50
   40 NSTRS=NSTRS+1
      GO TO 10
   50 PRINT 60, NSTRS,10HFILE TAPE3
   60 FORMAT(/I5,* STREET NUMBERS AND NAMES WERE READ FROM *,A10)
      IF (NSTRS .GT. 0) RETURN

C        NO DATA ON TAPE3. LOOK FOR STREET DATA ON CARDS.
      DO 80 I=1,301
      READ 70, NUMSTR(I),(NAMSTR(J,I),J=1,7)
   70 FORMAT(I5,5X,7A10)
      IF (EOF(5)) 90,80
   80 NSTRS=I
   90 PRINT 60, NSTRS,5HCARDS
      RETURN
      END
```

```
      FUNCTION HM(TIME)
C        LATEST CHANGES --
C        FEB 1. 1977. HJI.   CORRECTED MINUTES CALCULATION.
C        JAN 7. 1977.  HJI.  ORIGINAL VERSION.
      IH=TIME                   $     IM=60.*(TIME-IH)+0.5
      IF (IM .LT. 60) GO TO 5
      IM=0                      $     IH=IH+1
    5 ENCODE(10.10.HM) IH,IM
   10 FORMAT(I4.1H:.I2.2)
      RETURN
      END
```

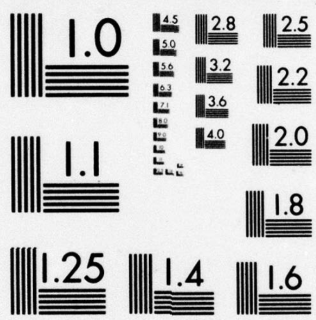
```
      SUBROUTINE  NUMBER(X,Y,HGT,NUM,ANG,FMT)

C         LATEST CHANGES  --
C         OCT 24, 1975. HJI.  ORIGINAL VERSION

      DIMENSION FORM(3),TEXT(3)
      DATA FORM/1H(,0,1H)/

      TEXT(1)=TEXT(2)=TEXT(3)=1H
      FORM(2)=FMT
      ENCODE (30,FORM,TEXT) NUM
      NC=30
      DO 10 I=1,3
      DO 10 J=6,60,6
      IF ((SHIFT(TEXT(4-I),6-J) .A. 77B) .NE. 55B) GO TO 20
   10 NC=NC-1
   20 CALL  SYMBOL(X,Y,HGT,TEXT,ANG,NC)
      RETURN
      END
```

```
      SUBROUTINE CUMTD(ISEG,CORT,NSG,DIS,TIM,NHT,RQ,SCOLL,TSTOPH,TSTOPR)

C        LATEST CHANGES --
C        MAR 10, 1977. HJI.  ADDED ARGUMENT RQ (TOTAL REFUSE QUANTITY).
C        JAN 19, 1977. HJI.  ADDED ARGUMENT NHT (TOTAL NUMBER OF HOUSES)
C        JAN 10, 1977.  HJI.  CRIGINAL VERSION.

      COMMON TITLE(8),NSEG,DUMMY(700,3),FLEN(700),NH(700),FMPH(700),
     1  NWAY(700),RQF(700)
      DIMENSION ISEG(100),CORT(100)

      NHT=0                    $      SUMD=SUMR=SUMT=0.
      DO 20 I=1,NSG
      J=ISEG(I)
      SUMD=SUMD+FLEN(J)
      IF (CORT(I) .NE. 1HC) GO TO 10
      SUMT=SUMT+60.*FLEN(J)/SCOLL+NH(J)*(TSTOPH+RQF(J)*TSTOPR)
      NHT=NHT+NH(J)            $      SUMR=SUMR+NH(J)*RQF(J)
      GO TO 20
   10 SUMT=SUMT+60.*FLEN(J)/FMPH(J)
   20 CONTINUE
      DIS=SUMD                 $      TIM=SUMT       $      RQ=SUMR
      RETURN
      END
```

AD-A060 987 NEW MEXICO UNIV ALBUQUERQUE ERIC H WANG CIVIL ENGINE--ETC F/G 13/2
AIR FORCE REFUSE-COLLECTION SCHEDULING PROGRAM DESCRIPTION. VOL--ETC(U)
JUL 78 H J IUZZOLINO, P STANS F29601-76-C-0015
UNCLASSIFIED CERF-EE-23 CEEDO-TR-78-23-VOL-4 NL

2 OF 2
AD
A060987

END
DATE
FILMED
1-79
DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
      SUBROUTINE SHLSRT(X,A,NW,SGN)                                     SHLS4010
      DIMENSION  X(1),   A(1)                                           SHLS4020
C                                                                       SHLS4030
C        THIS SUBROUTINE SORTS NW WORDS STARTING AT  X(1).              SHLS4040
C        IF SGN = 1..   X IS SORTED IN INCREASING ORDER.                SHLS4050
C        IF SGN =-1..   X IS SORTED IN DECREASING ORDER.                SHLS4060
C        ARRAY  A IS REORDERED AS  X IS SORTED SO THAT EACH  A          SHLS4070
C        ALWAYS CORRESPONDS TO THE SAME  X.                             SHLS4080
C                                                                       SHLS4090
      N=NW/2                                                            SHLS4100
   10 K=NW-N                                                            SHLS4110
      DO 50 I=1,K                                                       SHLS4120
      J=I                      $      L=I+N                             SHLS4130
      XT=X(L)                  $      AT=A(L)                           SHLS4140
   20 IF (SGN*(XT-X(J)) .GE. 0.) GO TO 40                               SHLS4150
      X(L)=X(J)                $      A(L)=A(J)                         SHLS4160
      L=J                      $      J=J-N     $     IF (J .GT. 0) GO TO 20SHLS4170
   40 X(L)=XT                  $      A(L)=AT                           SHLS4180
   50 CONTINUE                                                          SHLS4190
      IF (N .LE. 1) RETURN                                              SHLS4200
      N=(N+1)/2                $      GO TO 10                          SHLS4210
      END                                                               SHLS4220
```

```
      SUBROUTINE POSIT9(LSEQN,NSEQN)

C        LATEST CHANGES  --
C        JAN 24, 1977. HJI.   ORIGINAL VERSION.

C        POSIT9 POSITIONS TAPE9 IN FRONT OF TRIP NUMBER NSEQN.
C        LSEQN IS THE SEQUENCE NUMBER OF THE LAST TRIP READ FROM TAPE9.
C        THE TRIP FROM DUMP TO GARAGE IS TRIP NSEQN=0.

   10 IF (NSEQN-LSEQN-1) 20,70,50
   20 REWIND 9
      LSEQN=-1                  $      IF (NSEQN .EQ. 0) RETURN
      READ (9,30) KS
   30 FORMAT(I5)
      KSKP=KS/8+1
      READ (9,40) (DUMMY,K=1,KSKP)
   40 FORMAT(A10)
      LSEQN=0
      GO TO 10
   50 KLIM=3                    $      IF (LSEQN .LT. 0) KLIM=1
      DO 60 KK=1,KLIM
      READ (9,30) KS
      KSKP=KS/8+1
   60 READ (9,40) (DUMMY,K=1,KSKP)
      LSEQN=LSEQN+1
      GO TO 10
   70 RETURN
      END
```

```
      SUBROUTINE SHAPCOM(TOTLEN,AVMD,W,DI,DF,DIR)                       SPC40010
C                                                                       SPC40020
C        LATEST CHANGES  --                                             SPC40030
C        FEB 18, 1977. HJI.  ADDED ARGUMENT DIR.                        SPC40040
C        FEB 4, 1977. HJI.   ADAPTED FOR USE IN SCHEDULE GENERATING     SPC40050
C        PROGRAM (PHASE4).  ADDED ARGUMENTS W,DI,DF.  CHANGED /COPARM/  SPC40060
C        MAY 13, 1976. HJI.  ADAPTED FOR USE IN SECTIONING PROGRAM.     SPC40070
C        MAR 4, 1976.  HJI.  ADDED VALIDITY TEST FOR LENGTHS OF ANGLE   SPC40080
C        SIDES.                                                         SPC40090
C        OCT 28, 1975. HJI.  REMOVED 2 MORE BUGS IN S CURVE CODING      SPC40100
C        AND IMPROVED H CALCULATION BY IMPROVING RPR ESTIMATE.          SPC40110
C                                                                       SPC40120
      EQUIVALENCE (ISF,SF)                                              SPC40130
      COMMON /COPARM/ SF,XNI,XNF,YNI,YNF,SX,SY,RPR,C11,C12,XCTR,YCTR,   SPC40140
     1  BR1,BR2,     NPC,A(3),B(3),RATIO(3),XINT(3),YINT(3)             SPC40150
      DATA TWOPI/6.283185308/                                           SPC40160
                                                                        SPC40170
      NPC=1                   $        BR1=BR2=0.                       SPC40180
      DX=XNF-XNI              $        CY=YNF-YNI                       SPC40190
      SX=DX/TOTLEN            $        SY=DY/TOTLEN                     SPC40200
      A(1)=1.                 $        IF (DF .NE. 0.) A(1)=(DF-DI)/TOTLEN SPC40210
      B(1)=DI                 $        AW=ABS(W)                        SPC40220
      SGNW=1.                 $        IF (W .LT. 0.) SGNW=-1.          SPC40230
      XINT(1)=.5*(DX+SGNW*DY)+XNI $      YINT(1)=.5*(DY-SGNW*DX)+YNI    SPC40240
      RATIO(1)=1.-2.*AW/TOTLEN                                          SPC40250
      IF (ISF .EQ. 0  .OR.  ISF .EQ. 77B) RETURN                        SPC40260
                                                                        SPC40270
      THETA=ATAN2(DY,DX)                                                SPC40280
      D=SQRT(DX**2+DY**2)*AVMD                                          SPC40290
      IF (ISF.NE.2RRC .A. ISF.NE.2RLC .A. ISF.NE.2RRS .A. ISF.NE.2RLS)  SPC40300
     1      GO TO 40                                                    SPC40310
                                                                        SPC40320
C        PROCESS SHAPE FACTOR FOR CIRCULAR ARC OR S CURVE.              SPC40330
      XE=XNF                  $        YE=YNF                           SPC40340
      BR1=TOTLEN              $        CD=0           $      SGN=DIR    SPC40350
      A(1)=1.-2.*AW/BR1       $        B(1)=AW                          SPC40360
      IF (DF .EQ. 0.) GO TO 10                                          SPC40370
      A(1)=(DF-DI)/BR1        $        B(1)=DI                          SPC40380
   10 IF (ISF .EQ. 2RRC .OR.  ISF .EQ. 2RLC) GO TO 20                   SPC40390
```

```
      XE=0.5*(XNI+XNF)        $      YE=0.5*(YNI+YNF)                  SPC40400
      BR1=0.5*BR1             $      DD=0.5*DD      $    SGN=1.        SPC40410
      A(2)=A(1)               $      B(2)=0.                           SPC40420
      IF (DF .EQ. 0.) GO TO 20                                         SPC40430
      A(1)=1.-DI/BR1          $      A(2)=DF/BR1-1.                    SPC40440
   20 IF (ISF .EQ. 2RLC   .OR.  ISF .EQ. 2RLS) SGN=-SGN                SPC40450
      V=1.-D/TOTLEN           $      VS=V*V                            SPC40460
      ARG=V*(.6332067+VS*(.4303681+VS*(-.2629513+.1998384*VS)))        SPC40470
      IF (ARG .LE. 0.) GO TO 100                                       SPC40480
      RPR=SGN*TWOPI*SQRT(ARG)/BR1                                      SPC40490
C        IMPROVE APPROXIMATION OF RPR                                  SPC40500
      EPS1=SIN(.5*BR1*RPR)-.5*DD*RPR                                   SPC40510
      IF (ABS(EPS1) .LT. 1.E-5) GO TO 30                               SPC40520
      DRPR=SIGN(.0002,EPS1)                                            SPC40530
      EPS2=SIN(.5*BR1*(RPR+DRPR))-.5*DD*(RPR+DRPR)                     SPC40540
      RPR=RPR-EPS1*DRPR/(EPS2-EPS1)                                    SPC40550
   30 CONTINUE                                                         SPC40560
      R=1./RPR                $      ARG=R*R-0.25*DD*DD                SPC40570
      H=0.                    $      IF (ARG .GT. 0.) H=SQRT(ARG)/AVMD SPC40580
      IF (BR1 .GT. 3.14159*ABS(R)) H=-H                                SPC40590
      XINT(1)=XCTR=0.5*(XNI+XE)-SGN*SIN(THETA)*H                       SPC40600
      YINT(1)=YCTR=0.5*(YNI+YE)+SGN*COS(THETA)*H                       SPC40610
      XINT(2)=XNI+XNF-XCTR    $      YINT(2)=YNI+YNF-YCTR              SPC40620
      RATIO(1)=1.+W*RPR       $      RATIO(2)=2.-RATIO(1)              SPC40630
                                                                       SPC40640
C        SET UP ROTATION COEFFICIENTS                                  SPC40650
      C11=XNI-XCTR            $      C12=YNI-YCTR                      SPC40660
      RETURN                                                           SPC40670
                                                                       SPC40680
   40 IF (ISF .NE. 2RRR .AND. ISF .NE. 2RLR) GO TO 70                  SPC40690
      BR1=0.5*(TOTLEN-D)                                               SPC40700
      IF (BR1 .GT. 0.05*TOTLEN) GO TO 50                               SPC40710
      ISF=0                   $      RETURN                            SPC40720
   50 BR2=0.5*(TOTLEN+D)                                               SPC40730
      SX=SIN(THETA)/AVMD      $   SY=COS(THETA)/AVMD                   SPC40740
      F1=F2=0.     $      SGN=DIR  $      IF (ISF .EQ. 2RLR) SGN=-DIR  SPC40750
      IF (SGN*SGNW .LE. 0.) GO TO 60                                   SPC40760
      IF (BR1 .GT. 2.*AW) F1=2.*BR1/(BR1-2.*AW)                        SPC40770
      IF (  D .GT. 2.*AW) F2=2.*  D/(  D-2.*AW)                        SPC40780
```

```
60 A(1)=1.+(F1*AW-DI)/BR1 $      B(1)=DI
   A(2)=1.+2.*F2*AW/D     $      B(2)=-F2*AW
   A(3)=1.+F1*AW/BR1      $      B(3)=-F1*AW
   IF (DF .GT. 0.) A(3)=(DF-BR2+F1*AW)/D
   RATIO(1)=RATIO(3)=1.-2.*AW/BR1  $      RATIO(2)=1.-2.*AW/D
   XINT(1)=.5*SGN*BR1*(SX-SGNW*SY)+XNI
   YINT(1)=-.5*SGN*BR1*(SY+SGNW*SX)+YNI
   XINT(2)=.5*(DX+SGNW*DY)+XNI+SGN*SX*BR1
   YINT(2)=.5*(DY-SGNW*DX)+YNI-SGN*SY*BR1
   XINT(3)=.5*SGN*BR1*(SX+SGNW*SY)+XNF
   YINT(3)=-.5*SGN*BR1*(SY-SGNW*SX)+YNF
   IF (DIR .LT. 0.) ISF=36448-ISF
   RETURN

70 SGN=SIGN(1.,SF)*DIR    $      BR1=.5*(1.-DIR)*TOTLEN+DIR*ABS(SF)
   BR2=TOTLEN-BR1
   F=0.5*(1.-(BR2**2-BR1**2)/D**2)
   ARG=BR1**2-(F*D)**2    $      IF (ARG .LT. 0.00001) GO TO 100
   H=-SGN*SQRT(ARG)
   XCTR=XNI+(COS(THETA)*F*D-SIN(THETA)*H)/AVMD
   YCTR=YNI+(COS(THETA)*H+SIN(THETA)*F*D)/AVMD
   F1=F2=0.               $      IF (SGN*SGNW .LE. 0.) GO TO 90
   IF (BR1 .GT. 2.*AW) F1=BR1/(BR1-2.*AW)
   IF (BR2 .GT. 2.*AW) F2=BR2/(BR2-2.*AW)
90 A(1)=1.+(F1*AW-DI)/BR1      $      B(1)=DI
   A(2)=1.+F2*AW/BR2           $      B(2)=-F2*AW
   IF (DF .GT. 0.) A(2)=(DF-BR1+F2*AW)/BR2
   RATIO(1)=1.-2.*AW/BR1  $      RATIO(2)=1.-2.*AW/BR2
   XINT(1)=.5*(XCTR+XNI+SGNW*(YCTR-YNI))
   YINT(1)=.5*(YCTR+YNI-SGNW*(XCTR-XNI))
   XINT(2)=.5*(XNF+XCTR+SGNW*(YNF-YCTR))
   YINT(2)=.5*(YNF+YCTR-SGNW*(XNF-XCTR))
   RETURN

100 SF=BR1=BR2=0.         $      RETURN
   END
```

```
      SUBROUTINE COORD(XX,YY.CUMLEN,IERR)                               CORD4010
C                                                                       CORD4020
C        LATEST CHANGES  --                                             CORD4030
C        FEB 4. 1977. HJI.    ADAPTED TO GIVE DISPLACEMENT TO SIDE OF   CORD4040
C        LINE.  ADDED ARGUMENT IERR. MODIFIED /COPARM/ COMMON.          CORD4050
C        APR 14. 1975. HJI. BUG REMOVED FROM S-CURVE CALCULATION.       CORD4060
C        APR 9. 1975. HJI. ORIGINAL VERSION.                            CORD4070
                                                                        CORD4080
      COMMON /COPARM/ SF,XNI.XNF,YNI,YNF.SX,SY,RPR,C11,C12,XCTR,YCTR,   CORD4090
     1  BR1.BR2.      NPC,A(3).B(3),RATIO(3).XINT(3),YINT(3)            CORD4100
      INTEGER SF                                                        CORD4110
                                                                        CORD4120
      IERR=1                   $      IF (RATIO(NPC) .LT. 0.) RETURN    CORD4130
      IERR=0                                                            CORD4140
      S=CUMLEN                                                          CORD4150
      IF (SF  .NE. 0) GO TO 10                                          CORD4160
      S=A(1)*S+B(1)                                                     CORD4170
      XX=XNI+SX*S              $      YY=YNI+SY*S                       CORD4180
      GO TO 80                                                          CORD4190
   10 IF (SF.NE.2RRC .A. SF.NE.2RLC .A. SF.NE.2RRS .A. SF.NE.2RLS)      CORD4200
     1      GO TO 30                                                    CORD4210
C        GENERATE COORDINATES FOR CIRCLE OR S CURVE                     CORD4220
      RIP=RPR                                                           CORD4230
      XC=XCTR                  $      YC=YCTR                           CORD4240
      C1=C11                   $      C2=C12                            CORD4250
      IF (S .LE. .999*BR1 .OR. SF .EQ. 2RRC .OR. SF .EQ. 2RLC) GO TO 20 CORD4260
      RIP=-RPR                 $      S=S-BR1          $      NPC=2     CORD4270
      XC=XNI+XNF-XCTR          $      YC=YNI+YNF-YCTR                   CORD4280
      C1=0.5*(XNI+XNF)-XC      $      C2=0.5*(YNI+YNF)-YC               CORD4290
   20 S=A(NPC)*S+B(NPC)                                                 CORD4300
      SN=SIN(S*RIP)            $      CN=COS(S*RIP)                     CORD4310
      XX=C1*CN-C2*SN+XC        $      YY=C2*CN+C1*SN+YC                 CORD4320
      GO TO 80                                                          CORD4330
   30 IF (SF  .NE. 2RRR .AND. SF .NE. 2RLR) GO TO 60                    CORD4340
C        GENERATE COORDINATES FOR A RECTANGLE                           CORD4350
      SGN=1.                   $      IF (SF .EQ. 2RLR) SGN=-1.         CORD4360
      IF (S .GT. 1.05*BR1) GO TO 40                                     CORD4370
      IF (S .GT. 0.95*BR1) S=BR1                                        CORD4380
      S=A(1)*S+B(1)                                                     CORD4390
```

```
      XX=XNI+SGN*SX*S            $      YY=YNI-SGN*SY*S                  CORD4400
      GO TO 80                                                           CORD4410
   40 IF (S .GT. 1.05*BR2)  GO TO 50                                     CORD4420
      IF (S .GT. 0.95*BR2)  S=BR2                                        CORD4430
      S=S-BR1                    $      S=A(2)*S+B(2)       $     NPC=2  CORD4440
      XX=XNI+SGN*SX*BR1+SY*S            $     YY=YNI-SGN*SY*BR1+SX*S     CORD4450
      GO TO 80                                                           CORD4460
   50 S=S-BR2                    $      S=A(3)*S+B(3)       $     NPC=3  CORD4470
      XX=XNF+SGN*SX*(BR1-S)             $     YY=YNF-SGN*SY*(BR1-S)      CORD4480
      GO TO 60                                                           CORD4490
C        GENERATE COORDINATES FOR AN ANGLE                               CORD4500
   60 IF (S .GT. BR1)  GO TO 70                                          CORD4510
      S=A(1)*S+B(1)                                                      CORD4520
      XX=XNI+(XCTR-XNI)*S/BR1                                            CORD4530
      YY=YNI+(YCTR-YNI)*S/BR1                                            CORD4540
      GO TO 80                                                           CORD4550
   70 S=S-BR1                                                            CORD4560
      S=A(2)*S+B(2)              $      NPC=2                            CORD4570
      XX=XCTR+(XNF-XCTR)*S/BR2     $     YY=YCTR+(YNF-YCTR)*S/BR2        CORD4580
   80 XX=XINT(NPC)+(XX-XINT(NPC))*RATIO(NPC)                             CORD4590
      YY=YINT(NPC)+(YY-YINT(NPC))*RATIO(NPC)                             CORD4600
      RETURN                                                             CORD4610
      END                                                                CORD4620
```

```
      SUBROUTINE PRSCHED(NSC,ITRIP,NTPS,NSP,NNP,CORT,NSG,TC)

C         LATEST CHANGES  --
C         FEB 1, 1977. HJI.    ADDED VEHICLE CAPACITY TC TO ARGUMENT LIST.
C         JAN 29, 1977. HJI.   ORIGINAL VERSION

      COMMON TITLE(8),NSEG,NSTR(700),NN1(700),NN2(700),FLEN(700),
     1  NH(700),FMPH(700),NWAY(700),RQF(700),XMID(700),YMID(700),SF(700)
      COMMON /NODDATA/KNODES,NODNUM(500),NBS(500),XNOD(500),YNOD(500)
      COMMON /STREETS/ NSTRS,NUMSTR(300),NAMSTR(7,300),NONAME(7),
     1  NMFILL(7),NMGAR(7)
      COMMON /TIMES/ TSTOPH,TSTOPR,TUNLD,TMXTR,TMXDAY,TSTART,SCOLL,
     1  DLUNCH,TLUNCH,DBRK(2),TBRK(2)
      DIMENSION NSP(100,4),NNP(100,4),CORT(100,4),NSG(4)
      DATA MKLINE/50/

      JJI=1               $     IF (NSC .EQ. 0) JJI=4
      IF (ITRIP .GT. 1) GO TO 30
C         INITIALIZE ROUTE PRINTING
      PRINT 10, 1H1
   10 FORMAT(A1,*ACTION*,T87,*SPEED*,T98,*TIME      DISTANCE  HOUSEHOLDS
     1LOAD*/ T87,*(MPH)     (HR:MIN)     (MILES)   SERVICED    (PCT)*/)
      TOTLD=0.            $     NHTOT=0   $       TIM=TSTART
      PRINT 20, HM(TIM)
   20 FORMAT(* LEAVE GARAGE*,T95,A7)
      NLINES=3            $     JJF=3+1/NTPS
      GO TO 100

   30 CONTINUE
      PRINT 50, HM(TIM)
   50 FORMAT(* LEAVE LAND FILL*,T95,A7)
      NLINES=MOD(NLINES+1,MKLINE) $      IF (NLINES .EQ. 0) PRINT 10, 1H1
      JJF=4

  100 DO 300 JJ=JJI,JJF
      II=1
C        PROCESS CURRENT SEGMENT
  120 NMSG1=NSP(II,JJ)     $      NMST1=NSTR(NMSG1)
      LST1=IFIND(NMST1,NUMSTR,NSTRS)
```

```
      IF (LST1 .LT. 1  .OR.  LST1 .GT. NSTRS) LST1=301                  PRSC0400
      ACT=CORT(II,JJ)           $     OLDTIM=TIM                        PRSC0410
      DIS=0.                                                            PRSC0420
C        PROCESS PRINTING OF END OF SEGMENT                             PRSC0430
      IF (ACT .NE. 1HC) GO TO 140                                       PRSC0440
C        COLLECTION PROCESSING                                          PRSC0450
      ACTION1=10HPICK UP ON  $     ACTION2=10HBOTH SIDES                PRSC0460
      NHS=NH(NMSG1)             $     NFS=IABS(NHS)                     PRSC0470
      IF (NHS .LT. 0) ACTION2=10HRIGHT SIDE                             PRSC0480
      IF (NFS .EQ. 1) ACTION2=1H  $     ISPD1=SCOLL                     PRSC0490
      TIM=TIM+FLEN(NMSG1)/SCOLL+NFS*(TSTOPH+RQF(NMSG1)*TSTOPR)/60.      PRSC0500
      NHTOT=NHTOT+NFS           $     TOTLD=TOTLD+NFS*RQF(NMSG1)        PRSC0510
      GO TO 180                                                         PRSC0520
C        TRAVEL PROCESSING                                              PRSC0530
  140 ACTION1=10H  DRIVE ON  $     ACTION2=1H                           PRSC0540
      ISPD1=FMPH(NMSG1)         $     NFS=0                             PRSC0550
  160 TIM=TIM+FLEN(NMSG1)/FMPH(NMSG1)                                   PRSC0560
  180 DIS=DIS+FLEN(NMSG1)                                               PRSC0570
C        FIND CROSS STREET, IF ANY                                      PRSC0580
      LST2=301                                                          PRSC0590
      LNOD=IFIND(NNP(II+1,JJ),NODNUM,KNODES)                            PRSC0600
        DO 200 I=10,50,10                                               PRSC0610
        NUMS=SHIFT(NBS(LNOD),10-I) .A. 1777B                            PRSC0620
        IF (NUMS .EQ. NMSG1  .OR.  NUMS .EQ. 0) GO TO 200               PRSC0630
        IF (NSTR(NUMS) .EQ. NMST1) GO TO 200                            PRSC0640
        NMST2=NSTR(NUMS)        $     LST2=IFIND(NMST2,NUMSTR,NSTRS)    PRSC0650
        IF (LST2 .LT. 1  .OR. LST2 .GT. NSTRS) LST2=301                 PRSC0660
        IF (LST2 .NE. 301) GO TO 220                                    PRSC0670
  200 CONTINUE                                                          PRSC0680
  220 IF (II .EQ. NSG(3) .AND. JJ .EQ. 3) LST2=302                      PRSC0690
      IF (II .EQ. NSG(4) .AND. JJ .EQ. 4) LST2=303                      PRSC0700
      II=II+1                                                           PRSC0710
      IF (II .GT. NSG(JJ) .OR. ACT .EQ. 1HC) GO TO 240                  PRSC0720
      NMSG2=NSP(II,JJ)          $     NMST2=NSTR(NMSG2)                 PRSC0730
      ISPD2=FMPH(NMSG2)         $     NMSG1=NMSG2                       PRSC0740
      IF (NMST1 .EQ. NMST2 .AND. ISPD1 .EQ. ISPD2 .AND.                 PRSC0750
     1   CORT(II,JJ) .NE. 1HC) GO TO 160                                PRSC0760
                                                                        PRSC0770
  240 PRINT 250, ACTION1,ACTION2,(NAMSTR(I,LST1),I=1,3),(NAMSTR(I,LST2),PRSC0780
```

```
  1  I=1,3),ISPD1,HM(TIM),DIS,NFS,INT(100.*TOTLD/TC)                    PRSC0790
250 FORMAT(  2A11,1X,     3A10,* TO *,3A10,I3,4X,A7,F10.1,2I10.0)       PRSC0800
    NLINES=MOD(NLINES+1,MKLINE) $      IF (NLINES .EQ. 0) PRINT 10, 1H1PRSC0810
    IB=1                                                                PRSC0820
251 IF (OLDTIM .GE. TBRK(IB) .OR. TIM .LT. TBRK(IB)) GO TO 253          PRSC0830
    IF (DBRK(IB) .LE. 0.) GO TO 253                                     PRSC0840
    OLDTIM=TIM               $      TIM=TIM+DBRK(IB)/60.                PRSC0850
    PRINT 252, HM(OLDTIM),HM(TIM)                                       PRSC0860
252 FORMAT(* BREAK TIME*,T90,A7,* TO*,A7)                               PRSC0870
    NLINES=MOD(NLINES+1,MKLINE) $      IF (NLINES .EQ. 0) PRINT 10, 1H1PRSC0880
253 IF (IB .EQ. 2) GO TO 254                                            PRSC0890
    IB=2                     $      GO TO 251                           PRSC0900
254 IF (OLDTIM .GE. TLUNCH .OR. TIM .LT. TLUNCH) GO TO 256              PRSC0910
    IF (DLUNCH .LE. 0.) GO TO 256                                       PRSC0920
    OLDTIM=TIM               $      TIM=TIM+DLUNCH/60.                  PRSC0930
    PRINT 255, HM(OLDTIM),HM(TIM)                                       PRSC0940
255 FORMAT(* BREAK FOR LUNCH*,T90,A7,* TO*,A7)                          PRSC0950
    NLINES=MOD(NLINES+1,MKLINE) $      IF (NLINES .EQ. 0) PRINT 10, 1H1PRSC0960
256 CONTINUE                                                            PRSC0970
                                                                        PRSC0980
    IF (II .LE. NSG(JJ)) GO TO 120                                      PRSC0990
    IF (JJ .NE. 3   .OR.   TOTLD .LE. 0.) GO TO 300                     PRSC1000
    PRINT 260, HM(TIM),HM(TIM+TUNLD/60.)                                PRSC1010
260 FORMAT(*DUNLOAD*,T90,A7,* TO*,A7)                                   PRSC1020
    NLINES=MOD(NLINES+1,MKLINE) $      IF (NLINES .EQ. 0) PRINT 10, 1H1PRSC1030
    OLDTIM=TIM                                                          PRSC1040
    TIM=TIM+TUNLD/60.                                                   PRSC1050
    TOTLD=0.                                                            PRSC1060
    GO TO 251                                                           PRSC1070
300 CONTINUE                                                            PRSC1080
    RETURN                                                              PRSC1090
    END                                                                 PRSC1100
```

```
      SUBROUTINE STNAME(NSP,NNP,CORT,NSG)

C        LATEST CHANGE  --
C        FEB 25, 1977. HJI.   ORIGINAL VERSION.

      INTEGER SF
      COMMON TITLE(8),NSEG,NSTR(700),NN1(700),NN2(700),FLEN(700),
     1  NH(700),FMPH(700),NWAY(700),RQF(700),XMID(700),YMID(700),SF(700)
      COMMON /NODDATA/KNODES,NODNUM(500),NBS(500),XNOD(500),YNOD(500)
      COMMON /MAPDATA/ XMIN,XMAX,XLEN,YMIN,YMAX,YLEN,YHCUT,AVMD,WIDTH
      COMMON /COPARM/ISF,XNI,XNF,YNI,YNF,SX,SY,RPR,C11,C12,XCTR,YCTR,
     1  BR1,BR2,     NPC,A(3),B(3),RATIO(3),XINT(3),YINT(3)
      COMMON /STREETS/ NSTRS,NUMSTR(300),NAMSTR(7,300),OTHER(21)
      DIMENSION NSP(100,4),NNP(100,4),CORT(100,4),NSG(4)
      DIMENSION ICHAR(30),NSGTEM(20),FLTEM(20)
      EQUIVALENCE (XL,XMIN), (XR,XMAX), (YB,YMIN), (YT,YMAX)

      JF=3
      XMX=XLEN+1.
      XSC=XLEN/(XR-XL)          $      YSC=YLEN/(YT-YB)
      YCUT=YHCUT/YSC            $      SC=SQRT(XSC*YSC)

      DO 200  I=1,300
      NUMS=NUMSTR(I)            $      IF (NUMS .LE. 0) GO TO 210
      DECODE (30,10,NAMSTR(1,I)) ICHAR
   10 FORMAT(30A1)
C        FIND THE FIRST NONBLANK CHARACTER
        DO 20 J=1,30
        KI=J                    $      IF (ICHAR(J)  .NE. 1H ) GO TO 30
   20   CONTINUE
      GO TO 200
C        FIND THE LAST NONBLANK CHARACTER
   30   DO 40 J=1,30
        KF=31-J                 $      IF (ICHAR(KF)  .NE. 1H ) GO TO 50
   40   CONTINUE
   50 NCH=KF-KI+1               $      INS=1
      AHGT=1.6*WIDTH
      IF (AHGT .GE. 0.1) GO TO 55
      INS=0                     $      AHGT=0.1
```

```
   55 AWDTH=.90*NCH*AHGT                                                  STNM0400
                                                                          STNM0410
C        SAVE UP TO 20 SEGMENTS OF STREET NUMS                            STNM0420
      NSV=0                                                               STNM0430
        DO 60 J=1,NSEG                                                    STNM0440
        IF (NSTR(J) .NE. NUMS) GO TO 60                                   STNM0450
        NI=NN1(J)              $      NF=NN2(J)                           STNM0460
        XMD=XMID(J)            $      YMD=YMID(J)                         STNM0470
        LI=IFIND(NI,NODNUM,KNODES)     $      LF=IFIND(NF,NODNUM,KNODES)  STNM0480
        XNI=XNOD(LI)           $      YNI=YNOD(LI)                        STNM0490
        XNF=XNOD(LF)           $      YNF=YNOD(LF)                        STNM0500
        IF (XNI .LT. XL .OR. XNI .GT. XR .OR. YNI .LT. YB .OR. YNI .GT.   STNM0510
     1  YT) GO TO 60                                                      STNM0520
        IF (XMD .LT. XL .OR. XMD .GT. XR .OR. YMD .LT. YB .CR. YMD .GT.   STNM0530
     1  YT) GO TO 60                                                      STNM0540
        IF (XNF .LT. XL .OR. XNF .GT. XR .OR. YNF .LT. YB .OR. YNF .GT.   STNM0550
     1  YT) GO TO 60                                                      STNM0560
        NSV=NSV+1                                                         STNM0570
        NSGTEM(NSV)=J          $      FLTEM(NSV)=FLEN(J)                  STNM0580
        IF (SF(J) .EQ.0)FLTEM(NSV)=AVMD*SQRT((XNF-XNI)**2+(YNF-YNI)**2)   STNM0590
        IF (NSV .GE. 20) GO TO 70                                         STNM0600
   60   CONTINUE                                                          STNM0610
      IF (NSV .LT. 1) GO TO 200                                           STNM0620
C        SORT BY STREET LENGTH, LONGEST FIRST.                            STNM0630
   70 CALL SHLSRT(FLTEM,NSGTEM,NSV,-1.)                                   STNM0640
      FLMN=AVMD*AWDTH/SC                                                  STNM0650
      ISTR=0                   $      FLNSV=0.                            STNM0660
        DO 100 J=1,NSV                                                    STNM0670
        IF (FLMN .GT. FLTEM(J)) GO TO 110                                 STNM0680
C        SEE IF STREET IS TRAVELLED.                                      STNM0690
        NS=NSGTEM(J)                                                      STNM0700
          DO 80 K=1,JF                                                    STNM0710
          N=NSG(K)                                                        STNM0720
            DO 80 L=1,N                                                   STNM0730
            IF (NS .EQ. NSF(L,K)) GO TO 90                                STNM0740
   80       CONTINUE                                                      STNM0750
        ISTR=NS                $      FLNSV=FLTEM(J)      $      GO TO 120 STNM0760
   90   IF (ISTR .NE. 0) GO TO 100                                        STNM0770
        ISTR=NS                $      FLNSV=FLTEM(J)                      STNM0780
```

```
100   CONTINUE
110 INS=0
    IF (ISTR .EQ. 0) GO TO 200
C      APPEND THE STREET NAME TO SEGMENT ISTR
120 NI=NN1(ISTR)            $       NF=NN2(ISTR)   $      DIR=1.
    LI=IFIND(NI,NODNUM,KNODES)  $      LF=IFIND(NF,NODNUM,KNODES)
    IF (XNOD(LF)-XNOD(LI)) 140,130,150
130 IF (YNOD(LF)-YNOD(LI)   .GE. 0.) GO TO 150
140 LT=LI                   $      LI=LF          $      LF=LT
    DIR=-1.
150 XNI=XNOD(LI)            $      YNI=YNOD(LI)
    XNF=XNOD(LF)            $      YNF=YNOD(LF)
    ISF=SF(ISTR)
    AHGT=.7*AHGT
    W=(.5*AHGT+(1-INS)*(.5*AHGT+1.2*WIDTH))*AVMD/SC
    CALL SHAPCOM(FLNSV,AVMD,W,0.,0.,DIR)
    S=0.5*(FLNSV-FLMN)            $      DS=FLMN/NCH
    CALL COORD(XX,YY,S,IERR)      $      IF (IERR .GT. 0) GO TO 200
    NMAP=NMAPO=(YY-YB-.0001)/YCUT
    XPF=(XX-XL)*XSC+NMAP*XMX      $      YPF=(YY-YB-NMAP*YCUT)*YSC
C      PLOT INDIVIDUAL CHARACTERS
      DO 170 J=1,NCH
      XPI=XPF                     $      YPI=YPF
155   S=S+DS
      CALL COORD(XX,YY,S,IERR)    $      IF (IERR .GT. 0) GO TO 200
160   XPF=(XX-XL)*XSC+NMAP*XMX    $        YPF=(YY-YB-NMAP*YCUT)*YSC
      IF (SQRT((XPF-XPI)**2+(YPF-YPI)**2) .LT. 0.7*AHGT) GO TO 155
      ANG=57.295*ATAN2(YPF-YPI,XPF-XPI)
      CALL SYMBOL(XPI,YPI,AHGT,ICHAR(J+KI-1),ANG,1)
      NMAP=(YY-YB-.0001)/YCUT
      IF (NMAP .EQ. NMAPO) GO TO 170
      XPI=XPI+(NMAP-NMAPO)*XMX   $      YPI=YPI+(NMAPO-NMAP)*YHCUT
      NMAPO=NMAP                 $      GO TO 160
170   CONTINUE
200 CONTINUE
210 RETURN
    END
```

```
      SUBROUTINE MAPPLT(NRT,ITRIP,NTPS)                                 MPPL4010
                                                                        MPPL4020
C        LATEST CHANGES --                                              MPPL4030
C        FEB 8, 1977. HJI.    ADDED LEGEND PLOTTING.                    MPPL4040
C        FEB 4, 1977.  HJI.   ADDED STREET WIDTH CODING                 MPPL4050
C        JAN 31, 1977. HJI.   REMOVED SYMBOL AND NUMBER PLOTTING.       MPPL4060
C        CHANGED BLANK AND NODDATA COMMONS TO MATCH MAIN PROGRAM.       MPPL4070
C        JAN 14, 1977. HJI.   /MAPDATA/ COMMON CHANGED FOR USE IN PHASE4 MPPL4080
C        MAY 13, 1976. HJI.   ADAPTED FOR USE IN SECTIONING PROGRAM.    MPPL4090
                                                                        MPPL4100
      INTEGER SF                                                        MPPL4110
      COMMON TITLE(8),NSEG,NSTR(700),NN1(700),NN2(700),FLEN(700),       MPPL4120
     1 NH(700),FMPH(700),NWAY(700),RQF(700),XMID(700),YMID(700),SF(700)MPPL4130
      COMMON /NODDATA/KNODES,NODNUM(500),NBS(500),XNOD(500),YNOD(500)   MPPL4140
      COMMON /MAPDATA/ XMIN,XMAX,XLEN,YMIN,YMAX,YLEN,YHCUT,AVMD,WIDTH   MPPL4150
      COMMON /COPARM/ISF,XNI,XNF,YNI,YNF,SX,SY,RPR,C11,C12,XCTR,YCTR,   MPPL4160
     1 BR1,BR2,     NPC,A(3),B(3),RATIO(3),XINT(3),YINT(3)              MPPL4170
      COMMON /STREETS/ NSTRS,NUMSTR(300),NAMSTR(7,300),NONAME(7),       MPPL4180
     1 NMFILL(7),NMGAR(7)                                               MPPL4190
      DIMENSICN LEGEND(4)                                               MPPL4200
      DATA LEGEND/ 10HCOLLECT BO, 8HTH SIDES,  10HCOLLECT RI,8HGHT SIDE/MPPL4210
                                                                        MPPL4220
      XL=XMN=XMIN              $       YB=YMN=YMIN                      MPPL4230
      XR=XMAX                  $       YT=YMAX                          MPPL4240
      PHGT=YHCUT               $       XMX=XLEN+1.                      MPPL4250
      MX=YLEN     /PHGT+.99    $       PLEN=XMX*MX                      MPPL4260
      XSC=XLEN/(XMAX-XMN)      $       YSC=YLEN/(YMAX-YMN)              MPPL4270
      YCUT=PHGT/YSC                                                     MPPL4280
      JF=2                     $       IF (WIDTH .LE. 0.) JF=1          MPPL4290
      DO 210  J=1,JF                                                    MPPL4300
      W=(3-2*J)*WIDTH*AVMD/AMIN1(XSC,YSC)                               MPPL4310
      DO 200  K=1,NSEG                                                  MPPL4320
      NI=NN1(K)                $       NF=NN2(K)                        MPPL4330
      XMD=XMID(K)              $       YMD=YMID(K)                      MPPL4340
      ISF=SF(K)                $       IF (ISF .EQ. 778) GO TO 200      MPPL4350
      NS1=IFIND(NI,NODNUM,KNODES)  $     NS2=IFIND(NF,NODNUM,KNODES)    MPPL4360
      XNI=XNOD(NS1)            $       YNI=YNOD(NS1)                    MPPL4370
      XNF=XNOD(NS2)            $       YNF=YNOD(NS2)                    MPPL4380
      INBI=INBM=INBF=1                                                  MPPL4390
```

```
      IF (XNI .LT. XL .OR. XNI .GT. XR .OR. YNI .LT. YB .OR. YNI .GT.      MPPL4400
     1  YT) INBI=0                                                         MPPL4410
      IF (XMD .LT. XL .OR. XMD .GT. XR .OR. YMD .LT. YB .OR. YMD .GT.      MPPL4420
     1  YT) INBM=0                                                         MPPL4430
      IF (XNF .LT. XL .OR. XNF .GT. XR .OR. YNF .LT. YB .OR. YNF .GT.      MPPL4440
     1  YT) INBF=0                                                         MPPL4450
      IF (INBI .EQ. 0 .AND. INBM .EQ. 0 .AND. INBF .EQ. 0) GC TO 200       MPPL4460
      TOTLEN=FLEN(K)                                                       MPPL4470
      IF (SF( K) .EQ. 0) TCTLEN=AVMD*SQRT((XNF-XNI)**2+(YNF-YNI)**2)       MPPL4480
      NPMID=AMAX1(10.,1.+TOTLEN*XSC/AVMD,1.+TOTLEN*YSC/AVMD)               MPPL4490
      NPPSEG=2*NPMID                                                       MPPL4500
      CALL SHAPCOM(TOTLEN,AVMD,W,0.,0.,1.)                                 MPPL4510
      CUMLEN=0.               $      DS=TOTLEN/NPPSEG                      MPPL4520
      CALL COORD(XX,YY,CUMLEN,IERR)                                        MPPL4530
      NMAP=NMAPO=(YY-YMN-.0001)/YCUT                                       MPPL4540
      IPEN=3-INBI                                                          MPPL4550
      IF (IPEN .EQ. 3) GO TO 130                                           MPPL4560
      XP=(XX-XMN)*XSC+NMAP*XMX      $     YP=(YY-YMN-NMAP*YCUT)*YSC        MPPL4570
      CALL PLOT(XP,YP,3)                                                   MPPL4580
  130 DO 170 I=1,NPPSEG                                                    MPPL4590
      CUMLEN=CUMLEN+CS                                                     MPPL4600
      CALL COORD(XX,YY,CUMLEN,IERR)     $      IF (IERR .NE. 0) GO TO 200 MPPL4610
  140 XP= (XX-XMN)*XSC+NMAP*XMX                                            MPPL4620
      YP= (YY-YMN-NMAP*YCUT)*YSC                                           MPPL4630
      INB=1                                                                MPPL4640
      IF (XX.LT.XL .OR. XX.GT.XR .OR. YY.LT.YB .OR. YY.GT.YT) INB=0        MPPL4650
      IF ((IPEN .EQ. 3 .AND. INB .EQ. 0) .OR. NMAP .GE. MX) GO TO 160      MPPL4660
      CALL PLOT(XP,YP,IPEN)                                                MPPL4670
      IF (IPEN .EQ. 3) CALL FLOT(XP,YP,2)                                  MPPL4680
      IPEN=3-INB                                                           MPPL4690
  160 NMAP= (YY-YMN-.0001)/YCUT                                            MPPL4700
      IF (NMAP .EQ. NMAPO) GO TO 170                                       MPPL4710
      NMAPO=NMAP              $      IPEN=3          $      GO TO 140      MPPL4720
  170 CONTINUE                                                             MPPL4730
  200 CONTINUE                                                             MPPL4740
  210 CONTINUE                                                             MPPL4750
      CALL SYMBOL(0.,-.3,.2,TITLE,0.,40)                                   MPPL4760
      XP=XLEN-2.              $      YP=.05                                MPPL4770
      CALL PLOT(XP,YP,3)      $      CALL PLOT(XLEN,YP,2)                  MPPL4780
```

```
      CALL SYMBOL(XP,YP+.05,.12,6HTRAVEL,0.,6)                          MPPL4790
      DO 230 I=1,2                                                      MPPL4800
      XP=XLEN-2.            $      YP=YP-0.25                          MPPL4810
      CALL SYMBOL(XP,YP+.05,0.12,LEGEND(2*I-1),0.,20)                   MPPL4820
        DO 220 J=1,10                                                   MPPL4830
        IF (J .EQ. 1) CALL PLOT(XP,YP,3)                                MPPL4840
        CALL PLOT(XP+.1,YP,2)    $       IF (I .EQ. 1) GO TO 215        MPPL4850
        CALL PLOT(XP+.15,YP,3)   $       CALL PLOT(XP+.15,YP-.05,2)     MPPL4860
215     CALL PLOT(XP+.2,YP,3)                                           MPPL4870
220     XP=XP+.2                                                        MPPL4880
230   CONTINUE                                                          MPPL4890
300   CALL PLOT(PLEN+2.,0.,-3)                                          MPPL4900
      RETURN                                                            MPPL4910
      END                                                               MPPL4920
```

```
      SUBROUTINE PATHPLT(NSP,NNP,CORT,NSG,NTRIP,NTPS)

C        LATEST CHANGES  --
C        JUL 27, 1977. HJI.   REMOVED PEN DOWN ACROSS INTERSECTIONS ON
C        STRIP BOUNDARIES.
C        FEB 18, 1977. HJI.   MADE ARROWS LONGER.  ADDED DIR TO SHAPCOM.
C        FEB 8, 1977. HJI.    ORIGINAL VERSION.

      INTEGER SF
      LOGICAL RSO
      COMMON TITLE(8),NSEG,NSTR(700),NN1(700),NN2(700),FLEN(700),
     1  NH(700),FMPH(700),NWAY(700),RQF(700),XMID(700),YMID(700),SF(700)
      COMMON /NODDATA/KNODES,NODNUM(500),NBS(500),XNOD(500),YNOD(500)
      COMMON /MAPDATA/ XMIN,XMAX,XLEN,YMIN,YMAX,YLEN,YHCUT,AVMD,WIDTH
      COMMON /COPARM/ISF,XNI,XNF,YNI,YNF,SX,SY,RPR,C11,C12,XCTR,YCTR,
     1  BR1,BR2,     NPC,A(3),B(3),RATIO(3),XINT(3),YINT(3)
      DIMENSION NSP(100,4),NNP(100,4),CORT(100,4),NSG(4)
      DIMENSION ISEG(300),TRV(300),ITRV(300)
      EQUIVALENCE (ITRV,TRV)
      DATA KARO/30/

      ILAST=0
      JF=4                      $      IF (NTRIP .LT. NTPS) JF=3
      JF=3
      DO 20 J=1,JF
      N=NSG(J)
        DO 10 I=1,N
   10   TRV(I+ILAST)=ISEG(I+ILAST)=NSP(I,J)
   20 ILAST=ILAST+N
      CALL SHLSRT(TRV,ISEG,ILAST,1.)
      ITRV(1)=0
      DO 40 I=2,ILAST
   25 IF (ISEG(I-1) .LT. ISEG(I)) GO TO 40
C        REMOVE DUPLICATE SEGMENT NUMBER
      ILAST=ILAST-1
      IF (ILAST .LT. I) GO TO 40
        DO 30 J=I,ILAST
   30   ISEG(J)=ISEG(J+1)
      GO TO 25
```

```
40 ITRV(I)=0                                                             PAPL0400
                                                                         PAPL0410
   XL=XMIN $    XR=XMAX    $    YB=YMIN    $    YT=YMAX                  PAPL0420
   PHGT=YHCUT    $    XMX=XLEN+1.    $    MX=YLEN/PHGT+.99               PAPL0430
   XSC=XLEN/(XR-XL)         $     YSC=YLEN/(YT-YB)                       PAPL0440
   YCUT=PHGT/YSC            $     SC=SQRT(XSC*YSC)                       PAPL0450
   IFIRST=1                                                              PAPL0460
   NMAPO=-10                $     KTOT=0          $      XP=YP=0.        PAPL0470
   DO 160  J=1,JF                                                        PAPL0480
   NN=NSG(J)                                                             PAPL0490
     DO 160 I=1,NN                                                       PAPL0500
     NI=NNP(I,J)            $     NF=NNP(I+1,J)       $      KK=NSP(I,J) PAPL0510
     XMD=XMID(KK)           $     YMD=YMID(KK)   $      ISF=SF(KK)       PAPL0520
     LI=IFIND(NI,NODNUM,KNODES)      $      LF=IFIND(NF,NODNUM,KNODES)   PAPL0530
     IF (LI .GT. 0 .AND. LF .GT. 0) GO TO 60                             PAPL0540
     IF (LI .LT. 0) PRINT 50, J,I,LI                                     PAPL0550
50   FORMAT(*0---PIECE*,I3,I5,* TH NODE, NUMBER*,I5,* IS INCORRECT*/)PAPL0560
     IF (LF .LT. 0) PRINT 50, J,I+1,LF                                   PAPL0570
     GO TO 160                                                           PAPL0580
60   IF ((NN1(KK) .EQ. NI .AND. NN2(KK) .EQ. NF)  .OR.                   PAPL0590
  1       (NN1(KK) .EQ. NF .AND. NN2(KK) .EQ. NI)) GO TO 80              PAPL0600
     PRINT 70, J,I,KK                                                    PAPL0610
70   FORMAT(*0---PIECE*,I3,I5,* TH SEGMENT, NUMBER*,I5,* DOES NOT CONPAPL0620
 1NECT TO A BOUNDING NODE*/)                                             PAPL0630
     GO TO 160                                                           PAPL0640
80   CONTINUE                                                            PAPL0650
     XNI=XNOD(LI)           $     YNI=YNOD(LI)                           PAPL0660
     XNF=XNOD(LF)           $     YNF=YNOD(LF)                           PAPL0670
     INBI=INBM=INBF=1                                                    PAPL0680
     IF (XNI .LT. XL .OR. XNI .GT. XR .OR. YNI .LT. YB .OR. YNI .GT. PAPL0690
  1  YT) INBI=0                                                          PAPL0700
     IF (XMD .LT. XL .OR. XMD .GT. XR .OR. YMD .LT. YB .OR. YMD .GT. PAPL0710
  1  YT) INBM=0                                                          PAPL0720
     IF (XNF .LT. XL .OR. XNF .GT. XR .OR. YNF .LT. YB .OR. YNF .GT. PAPL0730
  1  YT) INBF=0                                                          PAPL0740
     IF (INBI .EQ. 0 .AND. INBM .EQ. 0 .AND. INBF .EQ. 0) GO TO 160      PAPL0750
     TOTLEN=FLEN(KK)                                                     PAPL0760
   IF (SF(KK) .EQ. 0) TOTLEN=AVMD*SQRT((XNF-XNI)**2+(YNF-YNI)**2)        PAPL0770
     NPMID=5.*SC*TOTLEN/AVMD+.5  $      IF (NPMID .LE. 0) NPMID=1        PAPL0780
```

```
      NPPSEG=2*NFMID-1       $      CUMLEN=0.      $      DS=TCTLEN/NPPSEG
      ISH=1                  $      IF (NI .NE. NN1(KK)) ISH=8
      DIR=1.                 $      IF (ISH .GT. 1) DIR=-1.
      LTR=IFIND(KK,ISEG,ILAST)  $      NTRV=(ITRV(LTR)/ISH) .A. 7
      W=(2.-NTRV)*WIDTH*AVMD/(3.*SC) $       ITRV(LTR)=ITRV(LTR)+ISH
      ACT=CORT(I,J)          $      RSO=NH(KK) .LT. 0
      CALL  SHAPCOM(TOTLEN,AVMD,W,0.,0.,DIR)
      CALL  COORD(XX,YY,CUMLEN,IERR)
      NMAP=(YY-YB-.0001)/YCUT
      IPEN=3-INBI
      IF (IPEN .EQ. 3  .OR.  IERR .GT. 0) GO TO 100
      XLAST=XP               $      YLAST=YP
      XP=(XX-XL)*XSC+NMAP*XMX       $      YP=(YY-YB-NMAP*YCUT)*YSC
      IF (IFIRST .EQ. 1 .OR. NMAP .NE. NMAPO) CALL PLOT(XP,YP,3)
      IFIRST=0
      NMAPO=NMAP             $      CALL PLOT(XP,YP,2)
100     DO 150 K=1,NPPSEG
        CUMLEN=CUMLEN+DS    $      KTOT=KTOT+1
        CALL COORD(XX,YY,CUMLEN,IERR)
        IF (IERR .NE. 0) GO TO 150
110     XLAST=XP            $      YLAST=YP
        XP=(XX-XL)*XSC+NMAP*XMX       $      YP=(YY-YB-NMAP*YCUT)*YSC
        INB=1
        IF (XX.LT.XL .OR. XX.GT.XR .OR. YY.LT.YB .OR. YY.GT.YT) INB=0
        IF ((IPEN .EQ. 3 .AND. INB .EQ. 0) .OR. NMAP .GE. MX)GO TO 140
        CALL PLOT(XP,YP,IPEN)         $      IFIRST=0
        IF (ACT .EQ. 1HC) GO TO 120
        IF (IPEN .EQ. 3) CALL PLOT(XP,YP,2)     $      IPEN=3-INB
        IF (MOD(KTOT,KARO) .EQ. 0) 125,140
120     IPEN=2
        IF (MOD(K,2) .EQ. 0) GO TO 130
        IPEN=3
        IF (MOD(KTOT,KARO) .GT. 1) GO TO 140
125     DX=.5*(XP-XLAST)    $      DY=.5*(YP-YLAST)
        CALL PLOT(XLAST-DY,YLAST+DX,2)
        CALL PLOT(XLAST+DY,YLAST-DX,2)
        CALL PLOT(XP,YP,2)
        GO TO 140
130     IF (.NOT. RSO) GO TO 140
```

```
      XM=.5*(XP+XLAST)      $      YM=.5*(YP+YLAST)                    PAPL1180
      X2=XM+YP-YM           $      Y2=YM-XP+XM                         PAPL1190
      CALL PLOT(XM,YM,3)         $      CALL PLOT(X2,Y2,2)             PAPL1200
  135 CALL PLOT(XP,YP,3)                                               PAPL1210
  140 NMAP=(YY-YB-.0001)/YCUT                                          PAPL1220
      IF (NMAP .EQ. NMAPO) GO TO 150                                   PAPL1230
      IF (MOD(KTOT,KARO) .LE. 1) KTOT=KTOT+2                           PAPL1240
      NMAPO=NMAP            $      IPEN=3          $      GO TO 110    PAPL1250
  150 CONTINUE                                                         PAPL1260
  160 CONTINUE                                                         PAPL1270
      RETURN                                                           PAPL1280
      END                                                              PAPL1290
```

```
      PROGRAM PHASE4(INPUT,OUTPUT,TAPE1,TAPE2,TAPE3=0,TAPE8,TAPE9,
     1  TAPE5=INPUT)

C       LATEST CHANGES --
C       JUL 7, 1977. HJI.    CORRECTED TREATMENT OF LUNCH ANC BREAK
C       TIMES IN ROUTE SUMMARY.
C       FEB 8, 1977. HJI.    ADDED PATH PLOTTING.
C       JAN 29, 1977. HJI.   ADDED CODING TO PRINT SCHEDULE.
C       JAN 19, 1977. HJI.   ADDED NHS ARRAY TO SECTION DATA.
C       JAN 11, 1976. HJI.   SEPARATED MAP BOUNDS AND SECTICN INFO INTO
C       TWO NONCORRESPONDING SETS OF ARRAYS.
C       JAN 4, 1977.  HJI.   ORIGINAL VERSION.

C       FILE ASSIGNMENTS --
C         TAPE 1 IS THE SEGMENT DATA         (BINARY)
C         TAPE 2 IS THE NODE DATA            (BINARY)
C         TAPE 3 IS THE STREET NAME DATA     (BUFFERED, PARITY=1)
C         TAPE 8 IS THE CALCOMP PLOT TAPE
C         TAPE 9 IS THE PATH DATA            (FORMATTED)

      COMMON TITLE(8),NSEG,NSTR(700),NN1(700),NN2(700),FLEN(700),
     1  NH(700),FMPH(700),NWAY(700),RQF(700),XMID(700),YMID(700),SF(700)
      COMMON /NODDATA/KNODES,NODNUM(500),NBS(500),XNOD(500),YNOD(500)
      COMMON /MAPDATA/ XMIN,XMAX,XLEN,YMIN,YMAX,YLEN,YHCUT,AVMD,WIDTH
      COMMON /STREETS/ NSTRS,NUMSTR(300),NAMSTR(7,300),NONAME(7),
     1  NMFILL(7),NMGAR(7)
      COMMON /TIMES/ TSTOPH,TSTOPR,TUNLD,TMXTR,TMXDAY,TSTART,SCOLL,
     1  DLUNCH,TLUNCH,DBRK(2),TBRK(2)
      COMMON /TEMSTG/ IBUFF(800)
      COMMON /U/ UNITS(2)
      DIMENSION CORT(100,4),DIST(4),NSG(4),NNP(100,4),NSP(100,4),
     1  TC(11),TMXTRV(10),VID(5,11)
      DIMENSION NSSRT(10) ,LV(50),IRS(50,2)
C       DIMENSIONS OF MAP BOUND DATA
      DIMENSION NSCN(100),NTRP(100),XMN(100),XMX(100),XLN(100),YMN(100),
     1  YMX(100),YLN(100)
C       DIMENSIONS OF SECTION INFORMATION
      DIMENSION DTIF(50,2),TTIF(50,2),TCAP(50),TLOAD(50),TEM(50),
     1  IORD(50),NHS(50)
```

```
      DATA NHS/50*0/
      DATA NTPS/0/, NONAME/7*1H /
      DATA NMFILL,NMGAR/ 9HLAND FILL,6*1H ,6HGARAGE,6*1H /
      DATA VID/50*1H , 10HNO VEHICLE, 10H SPECIFIED, 10H FOR THIS ,
     1  8HCAPACITY, 1H /
      DATA TC(11)/100000./
      DATA WIDTH/.10/

      READ 10, TITLE,UNITS,IPAIR
   10 FORMAT(8A10/2A10,I5)
      READ (1) NSEG,(NSTR(I),NN1(I),NN2(I),FLEN(I),NH(I),FMPH(I),
     1  NWAY(I),RQF(I),XMID(I),YMID(I),SF(I),I=1,NSEG),AVMD
      READ (2) NHTOT,TOTREF,KNODES,(NODNUM(I),NBS(I),XNOD(I),YNOD(I),
     1  I=1,KNODES)
      PRINT 20, TITLE,NSEG,KNOCES
   20 FORMAT(1H1,10X,8A10/*0DATA WERE READ FOR*,I5,* SEGMENTS AND*,I5,
     1 * NODES*)
      CALL STRIN

      READ 30, TSTOPH,TSTOPR,TUNLD,TMXTR,TMXDAY,TSTART,SCOLL,DLUNCH,
     1  TLUNCH,(DBRK(I),TBRK(I),I=1,2)
   30 FORMAT(7F10.0/6F10.0)
      PRINT 40, UNITS,TSTOPH,TSTOPR,TUNLD,TMXTR
   40 FORMAT(//*0INPUT DATA*/*0UNITS OF REFUSE QUANTITY=*,2A10/
     1  *0STOP TIME PER HOUSEHOLD =*,F10.2,* MINUTES*/
     2  *0STOP TIME PER UNIT REFUSE =*,F8.2, * MINUTES*/
     3  *0UNLOADING TIME =*,F10.2,* MINUTES*/
     4  *0MAXIMUM TRIP TIME =*,F7.2,* HOURS*)
      IF (TMXTR .GT. 0.) GO TO 50
      TMXTR=4.0
      PRINT 45, TMXTR
   45 FORMAT(1H+,32X,*, CHANGED TO *,F7.2,* HOURS BY PROGRAM*)
   50 PRINT 60, TMXDAY
   60 FORMAT(*0MAXIMUM ROUTE TIME =*,F6.2,* HOURS*)
      IF (TMXDAY .GT. 0.) GO TO 70
      TMXDAY=8.
      PRINT 45, TMXDAY
   70 PRINT 80, HM(TSTART)
   80 FORMAT(*0STARTING TIME =*,A7)
```

```
      IF (TSTART .GT. 0.) GO TO 90                                      PHS40790
      TSTART=8.00                                                       PHS40800
      PRINT 85, HM(TSTART)                                              PHS40810
   85 FORMAT(1H+,22X,*, CHANGED TO *,A7,* BY PROGRAM*)                  PHS40820
   90 PRINT 100, SCOLL                                                  PHS40830
  100 FORMAT(*0VEHICLE SPEED DURING COLLECTION =*,F7.0,* MPH*)          PHS40840
      IF (SCOLL .GT. 0.) GO TO 110                                      PHS40850
      SCOLL=5.                                                          PHS40860
      PRINT 105, SCOLL                                                  PHS40870
  105 FORMAT(1H+,44X,*, CHANGED TO *,F7.0, * MPH BY PROGRAM*)           PHS40880
  110 PRINT 120, DLUNCH,HM(TLUNCH),DBRK(1),HM(TBRK(1)),DBRK(2),         PHS40890
     1  HM(TBRK(2))                                                     PHS40900
  120 FORMAT(*0DURATION OF LUNCH =*F7.0,* MINUTES, STARTING AT ABOUT*,A7PHS40910
     1/ *0DURATION OF FIRST BREAK =*,F7.0,* MINUTES, STARTING AT ABOUT*,PHS40920
     2  A7/ *0DURATION OF SECOND BREAK =*,F6.0,* MINUTES, STARTING AT ABPHS40930
     3OUT*,A7)                                                          PHS40940
                                                                        PHS40950
C        READ VEHICLE CAPACITIES AND IDENTIFICATIONS                    PHS40960
      PRINT 130, UNITS                                                  PHS40970
  130 FORMAT(//*0VEHICLE INFORMATION*//5X,*CAPACITY*,8X,*IDENTIFICATION*PHS40980
     1  /2H (,2A10,1H)/)                                                PHS40990
      DO 160 I=1,11                                                     PHS41000
      READ 140, TC(I),(VID(J,I),J=1,5)                                  PHS41010
  140 FORMAT(F10.2,5A10)                                                PHS41020
      IF (EOF(5)) 175,145                                               PHS41030
  145 PRINT 150, TC(I),(VID(J,I),J=1,5)                                 PHS41040
  150 FORMAT(F13.2,5X,5A10)                                             PHS41050
  160 NTC=I                                                             PHS41060
                                                                        PHS41070
C        LOOK FOR MAP BOUNDS CARDS                                      PHS41080
  175 DO 190 I=1,101                                                    PHS41090
      READ 180, NSCN(I),NTRP(I),XMN(I),XMX(I),XLN(I),YMN(I),YMX(I),     PHS41100
     1  YLN(I)                                                          PHS41110
  180 FORMAT(2I5,6F10.0)                                                PHS41120
      IF (EOF(5)) 200,185                                               PHS41130
  185 IF (NTRP(I) .LT. 1) NTRP(I)=1                                     PHS41140
  190 NBC=I                                                             PHS41150
                                                                        PHS41160
  200 IF (NBC .GT. 0) GO TO 220                                         PHS41170
```

```
      PRINT 210
  210 FORMAT(///*0NO MAP BOUNDS WERE GIVEN FOR ANY TRIP.*/* EACH MAP WIL
     1L SHOW TRAVEL IN THE COLLECTION REGION BUT NOT NECESSARILY THE PAT
     2H TO OR FROM THE GARAGE OR LANDFILL.*)
      GO TO 260
  220 PRINT 230, NBC
  230 FORMAT(*1BOUNDS WERE SPECIFIED FOR*,I4,* MAPS*//   *0SECTION  TRIP
     1 XMIN       XMAX       XLEN        YMIN        YMAX        YLEN*/)
      PRINT 240, (NSCN(I),NTRP(I),XMN(I),XMX(I),XLN(I),YMN(I),YMX(I),
     1  YLN(I),I=1,NBC)
  240 FORMAT(I6,I7,F8.2,5F10.2)
      PRINT 250
  250 FORMAT(/*0WHERE NO BOUNDS WERE SPECIFIED, THE MAP WILL SHOW TRAVEL
     1 IN THE COLLECTICN*/* REGION BUT NOT NECESSARILY THE PATH TO OR FR
     2OM THE GARAGE OR LANDFILL.*)

  260 CONTINUE
      INPTU=9
  270 READ (INPTU,280) N,DIST(4),(NNP(I,4),NSP(I,4),CORT(I,4),I=1,N),
     1  NNP(N+1,4)
  280 FORMAT(I5,F10.0/8(I5,I4,A1))
      IF (EOF(INPTU)) 290,320
  290 IF (INFTU .EQ. 9) GO TO 310
      PRINT 300
  300 FORMAT(*0--- NO PATH DATA FOUND ON UNIT 9 OR ON CARDS ---*/*0--- J
     1OB TERMINATED ---*)
      CALL EXIT
  310 INPTU=5
      GO TO 270
  320 NSG(4)=N
      IF (INPTU .EQ. 9) GO TO 330
      REWIND 9
      WRITE (9,280) N,DIST(4),(NNP(I,4),NSP(I,4),CORT(I,4),I=1,N),
     1  NNP(N+1,4)
  330 CONTINUE

C        INITIALIZE FLOTTING
      CALL PLOTS(0,0,8)
      CALL PLOT(0.,-3.,3)   $     CALL PLOT(0.,0.,3)
```

```
      YHCUT= 30.                                                        PHS41570
                                                                        PHS41580
      CALL CUMTD(NSP(1.4),CORT(1,4),NSG(4),TOTD4,TOTT4,NHT,RG,SCOLL,    PHS41590
     1  TSTOPH.TSTOPR)                                                  PHS41600
                                                                        PHS41610
      PRINT 395                                                         PHS41620
  395 FORMAT(1H1,12X,*SECTION      SCALE         XMIN          XMAX         YMPHS41630
     1IN          YMAX*/)                                               PHS41640
      NSCOLD=NTRIP=0           $       INC=NBC                          PHS41650
      DO 480  I=1,100                                                   PHS41660
      TOTD=TOTT=TOTR=0.                                                 PHS41670
      XMIN=YMIN=1.E20          $       XMAX=YMAX=-1.E20                 PHS41680
        DO 420  J=1,3                                                   PHS41690
C         READ TRIP PATHS FROM UNIT INPTU                               PHS41700
        READ  (INPTU.400) N,DIST(J),NSC,TRC,TRL,(NNP(K,J),NSP(K,J),     PHS41710
     1    CORT(K,J),K=1,N),NNP(N+1,J)                                   PHS41720
  400   FORMAT(I5,F10.3,I5,2F10.3/ 8(I5,I4,A1))                         PHS41730
        IF (EOF(INPTU)) 490,410                                         PHS41740
  410   IF (INPTU .EQ. 5) WRITE (9,400) N,DIST(J),NSC,TRC,TRL,(NNP(K,J),PHS41750
     1    NSP(K,J),CORT(K,J),K=1,N),NNP(N+1,J)                          PHS41760
        CALL  CUMTD(NSP(1,J),CORT(1,J),N,DIS,TIM,NHT,RG,SCOLL,TSTOPH,   PHS41770
     1    TSTOPR)                                                       PHS41780
        TOTD=TOTD+DIS          $       TOTT=TOTT+TIM       $      TOTR=TOTR+RG PHS41790
        IF (J .EQ. 2) NHS(NSC)=NHT                                      PHS41800
  420   NSG(J)=N                                                        PHS41810
      NTRIP=NTRIP+1                                                     PHS41820
      IF (NSC .EQ. NSCOLD) GO TO 430                                    PHS41830
      NTRIP=1                  $       NSCOLD=NSC                       PHS41840
  430 IF (NTRIP .GT. NTPS) NTPS=NTRIP                                   PHS41850
      TCAP(NSC)=TRC            $       TLOAD(NSC)=TOTR                  PHS41860
      DTIF(NSC,NTRIP)=TOTD     $       TTIF(NSC,NTRIP)=TOTT             PHS41870
        DO 440  J=1,NBC                                                 PHS41880
        IF (NSCN(J) .EQ. NSC .AND. (NTRP(J) .EQ. NTRIP .OR. IPAIR .GT.  PHS41890
     1    0)) GO TO 480                                                 PHS41900
  440   CONTINUE                                                        PHS41910
  450 N=NSG(2)                 $       NP1=N+1                          PHS41920
        DO 460  J=1,N                                                   PHS41930
        X=XMID(NSP(J,2))       $       Y=YMID(NSP(J,2))                 PHS41940
        IF (X .EQ. 0.  .AND.  Y .EQ. 0.) GO TO 460                      PHS41950
```

```
      IF (XMAX .LT. X) XMAX=X    $      IF (XMIN .GT. X) XMIN=X          PHS41960
      IF (YMAX .LT. Y) YMAX=Y    $      IF (YMIN .GT. Y) YMIN=Y          PHS41970
460   CONTINUE                                                           PHS41980
      DO 470 J=1,NP1                                                     PHS41990
      LINE=IFIND(NNP(J,2),NODNUM,KNODES)                                 PHS42000
      X=XNOD(LINE)            $     Y=YNOD(LINE)                         PHS42010
      IF (X .EQ. 0.   .AND.  Y .EQ. 0.) GO TO 470                        PHS42020
      IF (XMAX .LT. X) XMAX=X    $      IF (XMIN .GT. X) XMIN=X          PHS42030
      IF (YMAX .LT. Y) YMAX=Y    $      IF (YMIN .GT. Y) YMIN=Y          PHS42040
470   CONTINUE                                                           PHS42050
     IF (NTRIP .EQ. 1) PRINT 475, 7HINITIAL,NSC,0.,XMIN,XMAX,YMIN,YMAX PHS42060
475  FORMAT(1X,A10,I6,4X,5F12.3)                                        PHS42070
     INC=INC+1                                                          PHS42080
     XLN(INC)=YLN(INC)=30.                                              PHS42090
     XLN(INC)=YLN(INC)=10.                                              PHS42100
     XMD=.5*(XMIN+XMAX)       $     YMD=.5*(YMIN+YMAX)                  PHS42110
     SC=XLN(INC)/(XMAX-XMIN)       $     YSC=YLN(INC)/(YMAX-YMIN)       PHS42120
     IF (YSC .LT. SC) SC=YSC                                            PHS42130
     IF (SC  .GT. 37.5*AVMD) SC=37.5*AVMD                               PHS42140
     IF (AVMD*WIDTH/SC .LT. .04) SC=0.8*SC                              PHS42150
     DX=XLN(INC)/SC           $     DY=YLN(INC)/SC                      PHS42160
     XMIN=XMD-.5*DX           $     XMAX=XMD+.5*DX                      PHS42170
     YMIN=YMD-.5*DY           $     YMAX=YMD+.5*DY                      PHS42180
     IF (NTRIP .EQ. 1) PRINT 475,8HADJUSTED,NSC,SC,XMIN,XMAX,YMIN,YMAX  PHS42190
     XMN(INC)=XMIN            $     XMX(INC)=XMAX                       PHS42200
     YMN(INC)=YMIN            $     YMX(INC)=YMAX                       PHS42210
     NSCN(INC)=NSC            $     NTRP(INC)=NTRIP                     PHS42220
480  CONTINUE                                                           PHS42230
490  IF (INPTU .EQ. 5) ENDFILE 9                                        PHS42240
     REWIND 9                                                           PHS42250
     PRINT 500,UNITS,((I,J,DTIF(I,J),TTIF(I,J),NHS(I),TCAP(I),TLOAD(I),PHS42260
   1  J=1,NTRIP),I=1,NSC)                                               PHS42270
500  FORMAT(*1SECTION   TRIP   DISTANCE     TIME   HOUSEHOLDS    CAPACITPHS42280
   1Y     LOAD*/ 19X,*(MILES)   (MINUTES)*,10X,1H(,2A10,1H)//           PHS42290
   2  (I8,I7,2F10.1,I10,3X,2F10.1))                                     PHS42300
                                                                        PHS42310
     IF (NTRIP .GT. 1) PRINT 510                                        PHS42320
510  FORMAT(*0ONLY HALF OF THESE TRIPS WILL BE USED.*)                  PHS42330
                                                                        PHS42340
```

```
      DO 530  I=1,NSC                                                   PHS42350
      TEM(I)=TCAP(I)                                                    PHS42360
  530 IORD(I)=I                                                         PHS42370
      IF (IPAIR .LT. 1) CALL SHLSRT(TEM,IORD,NSC,1.)                    PHS42380
                                                                        PHS42390
C        COUNT THE TRIPS FOR EACH CAPACITY VEHICLE.                     PHS42400
      ISTART=N=1              $      OLDTC=TCAP(IORD(1))                PHS42410
      TMX=TMXTR                                                         PHS42420
      DO 540  I=1,NSC                                                   PHS42430
      II=IORD(I)              $      TEM(I)=TTIF(II,1)/60.              PHS42440
      IF (TCAP(II) .EQ. OLDTC) GO TO 538                                PHS42450
      NSSRT(N)=I-ISTART       $      TMXTRV(N)=TMX                      PHS42460
      IF (NSSRT(N) .LE. 1) GO TO 534                                    PHS42470
      CALL SHLSRT(TEM(ISTART),IBUFF,NSSRT(N),1.)                        PHS42480
      TMX=TEM((NSSRT(N)+1)/2)                                           PHS42490
      IF (TMX .GT. TMXTRV(N)) TMXTRV(N)=TMX                             PHS42500
  534 IF (TMXTRV(N)  .GT. TMXTR) PRINT 536, TMXTRV(N),OLDTC             PHS42510
  536 FORMAT(*0THE MAXIMUM TRIP TIME WILL BE EXTENDED TO *,F6.2,* HOURS PHS42520
     1FOR VEHICLES OF CAPACITY *,F7.1/ * IF THIS IS UNSATISFACTORY, RERUPHS42530
     2N PROGRAMS PHASE2 AND PHASE3 WITH A SMALLER TIME LIMIT IN PHASE2*)PHS42540
      TMX=TMXTR                                                         PHS42550
      N=N+1                   $      ISTART=I       $      OLDTC=TCAP(II)   PHS42560
  538 IF (TTIF(II,1) .LE. 60.*TMX .OR. (NTRIP .GT. 1 .AND. TTIF(II,2)   PHS42570
     1   .LE. 60.*TMX)) GO TO 540                                       PHS42580
      TMX=TTIF(II,1)/60.                                                PHS42590
      IF (NTRIP .GT. 1 .AND. TTIF(II,2) .LT. 60.*TMX) TMX=TTIF(II,2)/60.PHS42600
  540 CONTINUE                                                          PHS42610
      TMXTRV(N)=TMX                                                     PHS42620
      IF (TMXTRV(N)  .GT. TMXTR) PRINT 536, TMX,OLDTC                   PHS42630
      NSSRT(N)=NSC-ISTART+1   $      NV=N                               PHS42640
                                                                        PHS42650
C        SELECT TRIPS AND PAIR THEM IF NTRIP=2.   PRINT THE RCUTE SUMMARYPHS42660
      PRINT 545, TITLE,UNITS,UNITS                                      PHS42670
  545 FORMAT(*1FINAL ROUTE SUMMARY*,10X,8A10// *0ROUTE      VEHICLE IDENTPHS42680
     1IFICATION*,25X,*VEHICLE   SECTION(S)   DISTANCE    TIME    HOUSEHOPHS42690
     2LDS   REFUSE*/ 58X,*CAPACITY  TRIP1 TRIP2   (MILES)  (HR:MIN)   SEPHS42700
     3RVICED   QUANTITY*/ 54X,1H(,A10,A5,1H),T108,1H(,A10,A5,1H))       PHS42710
      IF=0                    $      NRT=1                              PHS42720
      NHTOT=TOTDIS=TOTREF=TOTTIM=0.                                     PHS42730
```

```
      DO 620 I=1,NV                                                     PHS42740
      II=IF+1               $        IF=II+NSSRT(I)-1                   PHS42750
      T=TCAP(IORD(II))                                                  PHS42760
C        FIND THE LINE CONTAINING THE VEHICLE IDENTIFICATION.           PHS42770
      LVID=11                                                           PHS42780
        DO 550 J=1,NTC                                                  PHS42790
        IF (T .NE. TC(J)) GO TO 550                                     PHS42800
        LVID=J              $        GO TO 560                          PHS42810
  550   CONTINUE                                                        PHS42820
                                                                        PHS42830
  560 NI=NSSRT(I)                                                       PHS42840
      IF(NI .EQ. 1 .OR. NTRIP .EQ. 1) GO TO 580                         PHS42850
        DO 570 J=II,IF                                                  PHS42860
        IOJ=IORD(J)         $        TEM(J)=DTIF(IOJ,1)-DTIF(ICJ,2)     PHS42870
        IF (TTIF(IOJ,1) .GT. 60.*TMXTRV(I)) TEM(J)=TEM(J)+2000.         PHS42880
        IF (TTIF(IOJ,2) .GT. 60.*TMXTRV(I)) TEM(J)=TEM(J)-2000.         PHS42890
  570   CONTINUE                                                        PHS42900
      CALL SHLSRT(TEM(II),IORD(II),NI,1.)                               PHS42910
  580 LIM=(NI+NTRIP-1)/NTRIP                                            PHS42920
C        REPLACE THE DISTANCE SAVING BY THE APPROPRIATE TRIP TIME       PHS42930
        DO 585 J=1,NI                                                   PHS42940
  585   TEM(II+J-1)=TTIF( IORD(II+J-1), (J+LIM-1)/LIM )                 PHS42950
      IF (LIM .GT. 1 .AND. NTRIP .GT. 1) CALL SHLSRT(TEM(II),IORD(II),  PHS42960
     1 LIM,1.)                                                          PHS42970
      IF (NI-LIM .GT. 1)CALL SHLSRT(TEM(II+LIM),IORD(II+LIM),NI-LIM,1.) PHS42980
      LIM=(LIM+IPAIR)/(IPAIR+1)                                         PHS42990
        DO 610 J=1,LIM                                                  PHS43000
        NS1=IORD(II+J-1)    $        NS2=0                              PHS43010
        IF (IPAIR .GT. 0) NS1=IORD(II+2*J-2)                            PHS43020
        NHT=NHS(NS1)        $        REF=TLOAD(NS1)                     PHS43030
        DIS=DTIF(NS1,1)+TOTD4                                           PHS43040
        TMX=TIM=TTIF(NS1,1)+TOTT4+TUNLD                                 PHS43050
        IF (TIM .GE. TLUNCH-TSTART) TIM=TIM+DLUNCH/60.                  PHS43060
        IF (TBRK(1) .GT. TSTART .AND. TIM .GE. TBRK(1)-TSTART)          PHS43070
     1    TIM=TIM+DBRK(1)/60.                                           PHS43080
        IF (TBRK(2) .GT. TSTART .AND. TIM .GE. TBRK(2)-TSTART)          PHS43090
     1    TIM=TIM+DBRK(2)/60.                                           PHS43100
        IF (NTRIP+IPAIR .EQ. 1 .OR. 2*J .EQ. NI+1) GO TO 590            PHS43110
        NS2=IORD(IF+1-J)    $        IF (IPAIR .GT. 0) NS2=IORD(II+2*J-1) PHS43120
```

```
      NHT=NHT+NHS(NS2)       $     REF=REF+TLOAD(NS2)
      DIS=DIS+DTIF(NS2,2-IPAIR) $      TIM=TIM+TTIF(NS2,2-IFAIR)+TUNLD
      IF (TMX .LT. TLUNCH-TSTART .AND. TIM .GE. TLUNCH-TSTART)
     1  TIM=TIM+DLUNCH/60.
      IF (TMX .LT. TBRK(1)-TSTART .AND. TIM .GE. TBRK(1)-TSTART)
     1  TIM=TIM+DBRK(1)/60.
      IF (TMX .LT. TBRK(2)-TSTART .AND. TIM .GE. TBRK(2)-TSTART)
     1  TIM=TIM+DBRK(2)/60.
  590 PRINT 600, NRT,(VID(K,LVID),K=1,5),T,NS1,NS2,DIS,HM(TIM/60.),
     1 NHT,REF
  600 FORMAT(1H0,I4,4X,5A10,F6.1,2I6.0,F10.1,3X,A7,I10,F12.1)
      IRS(NRT,1)=NS1       $     IRS(NRT,2)=NS2      $    LV(NRT)=LVID
      TOTDIS=TOTDIS+DIS    $     TOTTIM=TOTTIM+TIM
      TOTREF=TOTREF+REF    $     NHTOT=NHTOT+NHT
  610 NRT=NRT+1
  620 CONTINUE
      PRINT 630, TOTDIS,HM(TOTTIM/60.),NHTOT,TOTREF
  630 FORMAT(///82X,*=====      ======       ====      ======*/ 74X,
     1 *TOTALS*,F7.1,3X,A7,I10,F12.1)
      NRT=NRT-1

      REWIND 9             $     LSEQN=-1       $     NRTP1=NRT+1
      DO 680 I=1,NRTP1
      JTRIPS=NTPS+IPAIR    $     IF (IRS(I,2) .EQ. 0) JTRIPS=1
      DO 670 J=1,JTRIPS
      NSC=0
      IF (I .EQ. NRTP1) GO TO 655
      NSC=IRS(I,J)         $     ISEQN=(NSC-1)*NTPS+J
      IF (J .GT. NTPS) ISEQN=NSC
      CALL POSIT9(LSEQN,ISEQN)
        DO 640 K=1,3
        READ (9,280) N,DIST(K),(NNP(L,K),NSP(L,K),CORT(L,K),L=1,N),
     1  NNP(N+1,K)
  640   NSG(K)=N
      LSEQN=LSEQN+1
      LVID=LV(I)
      IF (J .EQ. 1) PRINT 650, I,(TITLE(K),K=1,5),(VID(K,LVID),K=1,5)
  650 FORMAT(*1ROUTE*,I3,10X,5A10,5X,5A10)
      CALL PRSCHED(NSC,J,JTRIPS,NSP,NNP,CORT,NSG,TC(LVID))
```

```
  655     DO 660 K=1,IAC                                                    PHS43520
C       SCAN THE MAP BOUND DATA FOR SECTION NSC, TRIP J.  EACH TIME         PHS43530
C       BOUNDS ARE FOUND, DRAW A MAP.                                       PHS43540
          IF (NSCN(K) .NE. NSC .OR. (NTRP(K) .NE. J .AND. IPAIR .EQ. 0))   PHS43550
     1      GO TO 660                                                       PHS43560
          XMIN=XMN(K)    $      XMAX=XMX(K)    $      XLEN=XLN(K)           PHS43570
          YMIN=YMN(K)    $      YMAX=YMX(K)    $      YLEN=YLN(K)           PHS43580
          IF (XMIN .GE. XMAX .OR.  YMIN .GE. YMAX) GO TO 670                PHS43590
          IF (I .LT. NRTP1) CALL PATHPLT(NSP,NNP,CORT,NSG,J,JTRIPS)         PHS43600
          XP=0.5*XLEN                                                       PHS43610
          CALL NUMBER(XP,-.2,0.12,I,0.,10H5HROUTE,I3)                       PHS43620
          IF (JTRIPS .GT. 1)CALL NUMBER(XP+1.,-.2,0.12,J,0.,9H4HTRIP,I2)   PHS43630
          CALL SYMBOL(XP,-0.4,0.12,VID(1,LVID),0.,MIN1(50.,5.*XLEN-20.))   PHS43640
          CALL STNAME(NSP,NNP,CORT,NSG)                                     PHS43650
          CALL MAPPLT(I,J,JTRIPS)                                           PHS43660
  660     CONTINUE                                                          PHS43670
  670   CONTINUE                                                            PHS43680
  680 CONTINUE                                                              PHS43690
                                                                            PHS43700
      CALL PLOT(0.,0.,999)                                                  PHS43710
      CALL EXIT                                                             PHS43720
      END                                                                   PHS43730
```

APPENDIX C

DEFINITIONS OF IMPORTANT VARIABLES

Note: A single variable symbol may have different meanings in relation to the various subroutines. For this reason, variables are defined below for each subprogram and for the main program.

FUNCTION IFIND

IARRAY	Array being searched.
LEN	Length of IARRAY.
NUM	Number being sought.

SUBROUTINE STRIN

NAMSTR	Array of street names.
NSTRS	Count of street-name cards.
NUMSTR	Array of street numbers.

FUNCTION HM

IH	Integer number of hours in TIME.
IM	Number of minutes.
TIME	Time, in decimal hours.

SUBROUTINE NUMBER

FORM	Output format for number.
NUM	Number to be plotted.
TEXT	Character representation of number.

SUBROUTINE CUMTD

CORT	Array of collection-or-travel indicators.
DIS	Length of path, in miles.
FLEN	Array of segment lengths, in miles.
FMPH	Array of segment speed limits.
ISEG	Array of numbers of segments in path.
NH	Array of segment house counts.
NHT	Total house count for path.
NSG	Count of segments in path.
RQ	Total refuse quantity for path.
RQF	Array of segment refuse-quantity adjustment factors.

SCOLL	Vehicle speed during collection.
TIM	Time required to traverse path, in minutes.
TSTOPH	Stop time per collection location, in minutes.
TSTOPR	Stop time per unit refuse, in minutes.

SUBROUTINE SHLSRT

A	Array reordered as X is sorted.
NW	Count of words to be sorted.
SGN	If SGN = 1.0, X is sorted into increasing order. If SGN = -1.0, X is sorted into decreasing order.
X	Array to be sorted.

SUBROUTINE POSIT9

KLIM	Count of pieces in trip.
KS	Count of segments in piece.
KSKP	Count of card images to be skipped.
LSEQN	Sequence number of last trip read from TAPE9.
NSEQN	Sequence number of trip being sought on TAPE9.

SUBROUTINE SHAPCOM

A	Array giving slopes of linear mappings of segment pieces.
AVMD	Map distance conversion, in miles per MCU.
AW	Absolute value of street half-width, in miles.
B	Array of constants in the linear mappings of segment pieces.
BR1	Distance to first break in segment shape, in miles.
BR2	Distance to second break in segment shape, in miles.
DF	Displacement of end point of segment mapping, in miles.
DI	Displacement of starting point of segment mapping, in miles.
DIR	If DIR = 1.0, coordinates are generated from the starting to the ending node. If DIR = -1.0, coordinates are generated from the ending to the starting node.
DX	X-component of segment span, in MCU.
DY	Y-component of segment span, in MCU.
ISF	Shape code when in character form.
RATIO	Array of ratios of side length to segment length for segment pieces.

RPR	Reciprocal of radius of curvature for circular segments or circular portions of S-curves.
SF	Shape code when in binary form.
THETA	Slope of line from starting to ending node, in radians.
TOTLEN	Segment length, in miles.
W	Half-width of segment, in miles.
XNF,YNF	Coordinates of ending node.
XNI,YNI	Coordinates of starting node.

SUBROUTINE COORD

A	Array of slopes of linear mappings of segment pieces.
B	Array of constants in the linear mappings of segment pieces.
BR1	Distance to first break in segment shape, in miles.
BR2	Distance to second break in segment shape, in miles.
CUMLEN	Cumulative length along segment, in miles.
NPC	Number of piece of segment.
RATIO	Array of ratios of side length to segment length for segment pieces.
RPR	Reciprocal of radius of curvature for circular segments or circular portions of S-curves.
S	Linearly mapped distance along segment since previous break.
SF	Shape code.
XINT,YINT	Arrays of coordinates used in linear interpolations for segment coordinates.
XNF,YNF	Coordinates of ending node.
XNI,YNI	Coordinates of starting node.

SUBROUTINE PRSCHED

ACT	Collection-or-travel indicator for current segment.
CORT	Array of collection-or-travel indicators for the segments in the path.
DBRK	Array of durations of break times, in minutes.
DIS	Distance traveled since last line of printed output, in miles.
DLUNCH	Duration of lunch break, in minutes.
FLEN	Array of segment lengths, in miles.
FMPH	Array of segment speed limits, in mph.
KNODES	Count of nodes in map description.

MXLINE	Count of lines per page before page eject.
NAMSTR	Array of street names.
NFS	Count of stops since previous line of printed output.
NH	Array of segment house counts.
NMSG1	Number of current segment.
NMSG2	Number of next segment in path.
NNP	Array of numbers of nodes in path.
NSG	Array of counts of segments in path pieces.
NSP	Array of numbers of segments in path.
NSTR	Array of street numbers of segments.
NSTRS	Count of street names.
NTPS	Count of trip choices available per section.
NUMSTR	Array of street numbers.
OLDTIM	Time at completion of previous segment.
RQF	Array of segment refuse-quantity adjustment factors.
TBRK	Array of approximate break starting times.
TIM	Time at end of current segment.
TLUNCH	Approximate starting time of lunch break.
TOTLD	Cumulative vehicle load.
TSTART	Route starting time, in hours.
TSTOPH	Stop time per collection location, in minutes.
TSTOPR	Stop time per unit of refuse, in minutes.
TUNLD	Unloading time at the landfill, in minutes.

SUBROUTINE STNAME

AHGT	Height of the lettering, in inches.
AVMD	Map distance conversion, in miles per MCU.
AWDTH	Width of the street name, in inches.
CORT	Array of collection-or-travel indicators for segments in path.
FLTEM	Array of segment lengths, in miles.
ICHAR	Array of characters in street name.
INS	If INS = 0, street name is plotted along outside of street; if INS = 1, street name is plotted within street bounds.
KNODES	Count of nodes in map description.
NAMSTR	Array of street names.
NCH	Count of characters in street name.

NMAP	Map-strip number of the current point.
NMAPO	Map-strip number of the previous point.
NN1	Array of segment starting node numbers.
NN2	Array of segment ending node numbers.
NODNUM	Array of node numbers.
NSEG	Count of segments in map description.
NSG	Array of counts of segments in each piece of the trip.
NSP	Array of numbers of segments in trip.
WIDTH	Half-width of the street, in inches.
XL,XR	X-coordinates of left and right map boundaries, in MCU.
XNF,YNF	Coordinates of ending node, in MCU.
XNI,YNI	Coordinates of starting node, in MCU.
XNOD,YNOD	Arrays of node coordinates.
XPF,YPF	Plotter coordinates of start of next character in street name.
XPI,YPI	Plotter coordinates of start of present character in street name.
YB,YT	Y-coordinates of bottom and top map boundaries, in MCU.
YHCUT	Height of map strips, in inches.

SUBROUTINE MAPPLT

AVMD	Map distance conversion, in miles per MCU.
CUMLEN	Cumulative length along segment, in miles.
FLEN	Array of segment lengths, in miles.
INB	Point within map-bounds indicator.
ISF	Shape code when in character form.
KNODES	Count of nodes in map description.
NMAP	Map-strip number of the current point.
NMAPO	Map-strip number of the previous point.
NN1	Array of starting node numbers.
NN2	Array of ending node numbers.
NODNUM	Array of node numbers.
NPPSEG	Count of points plotted per segment.
PHGT	Height of map strip, in inches.
PLEN	Total length of plot, in inches.
SF	Shape code when in binary form.
TOTLEN	Segment length, in miles.
W	Half-width of street, in miles.

WIDTH	Half-width of street, in plotter inches.
XL,XR	X-coordinates of left and right map boundaries, in MCU.
XLEN	X-direction map length, in inches.
XNOD,YNOD	Arrays of node coordinates, in MCU.
YB,YT	Y-coordinates of bottom and top map boundaries, in MCU.
YCUT	Height of map strips, in MCU.
YLEN	Y-direction map length, in inches.

SUBROUTINE PATHPLT

ACT	Collection-or-travel indicator for current segment.
AVMD	Map distance conversion, in miles per MCU.
CUMLEN	Cumulative length along segment, in miles.
FLEN	Array of segment lengths, in miles.
INB	Segment-in-bounds indicator.
ISEG	Array of numbers of segments in path.
ITRV	Array of counts of times segments are traversed.
KARO	Count of steps before an arrowhead is plotted.
KK	Number of current segment.
NMAP	Map-strip number of current point.
NMAPO	Map-strip number of previous point.
NNP	Array of numbers of nodes in path.
NPPSEG	Count of points plotted per segment.
NSG	Array of counts of segments in pieces of trip.
NSP	Array of numbers of segments in path.
PHGT	Height of map strip, in inches.
RSO	Indicator for collection from only right side.
SF	Shape code.
TOTLEN	Segment length, in miles.
W	Half-width of street, in miles.
WIDTH	Half-width of street, in plotter inches.
XL,XR	X-coordinates of left and right map boundaries, in MCU.
XLEN,YLEN	X- and y-direction map lengths, in inches.
XNOD,YNOD	Arrays of node coordinates, in MCU.
YB,YT	Y-coordinates of bottom and top map boundaries, in MCU.
YCUT	Height of map strips, in MCU.

PROGRAM PHASE4

AVMD	Map distance conversion, in miles per MCU.
CORT	Array of collection-or-travel indicators.
DBRK	Array of break durations, in minutes.
DIST	Array of distances to section, within section, to landfill from section, and from landfill to garage.
DLUNCH	Duration of lunch break, in minutes.
DTIF	Array of travel distances to section, within section, and from section to landfill, for all sections.
INPTU	Number of input unit containing route data.
IORD	Array of pointers to trips.
IPAIR	Consecutive trip-pairing indicator.
IRS	Array of section numbers in order of output.
KNODES	Count of nodes in map description.
LV	Array of vehicle-identification line numbers.
NBC	Count of map-bounds cards.
NHS	Array of house counts per section.
NNP	Array of numbers of nodes in path.
NRT	Route counter.
NSC	Section number.
NTC	Count of vehicle identifications.
NTPS	Count of trip choices per section.
NRTP	Array of trip indicators from map-bounds cards.
SC	Map scale, in inches per MCU.
SCOLL	Vehicle speed during collection.
TBRK	Array of approximate break starting times, in hours.
TC	Array of vehicle capacities from vehicle-identification cards.
TCAP	Array of capacities of vehicles servicing sections.
TLOAD	Array of vehicle loads.
TLUNCH	Approximate starting time of lunch break, in hours.
TMXDAY	User-specified maximum route time per day.
TMXTR	User-specified maximum trip time, in hours.
TSTART	Route starting time, in hours.
TSTOPH	Stop time per collection location, in minutes.
TSTOPR	Stop time per unit of refuse, in minutes.
TTIF	Array of travel times to section, within section, and from section to landfill, for all sections.

TUNLD	Unloading time at the landfill, in minutes.
UNITS	Refuse unit.
VID	Array of vehicle identifications.
WIDTH	Half-width of street, in plotter inches.
XNOD,YNOD	Arrays of node coordinates, in MCU.

(The reverse of this page is blank.)

APPENDIX D

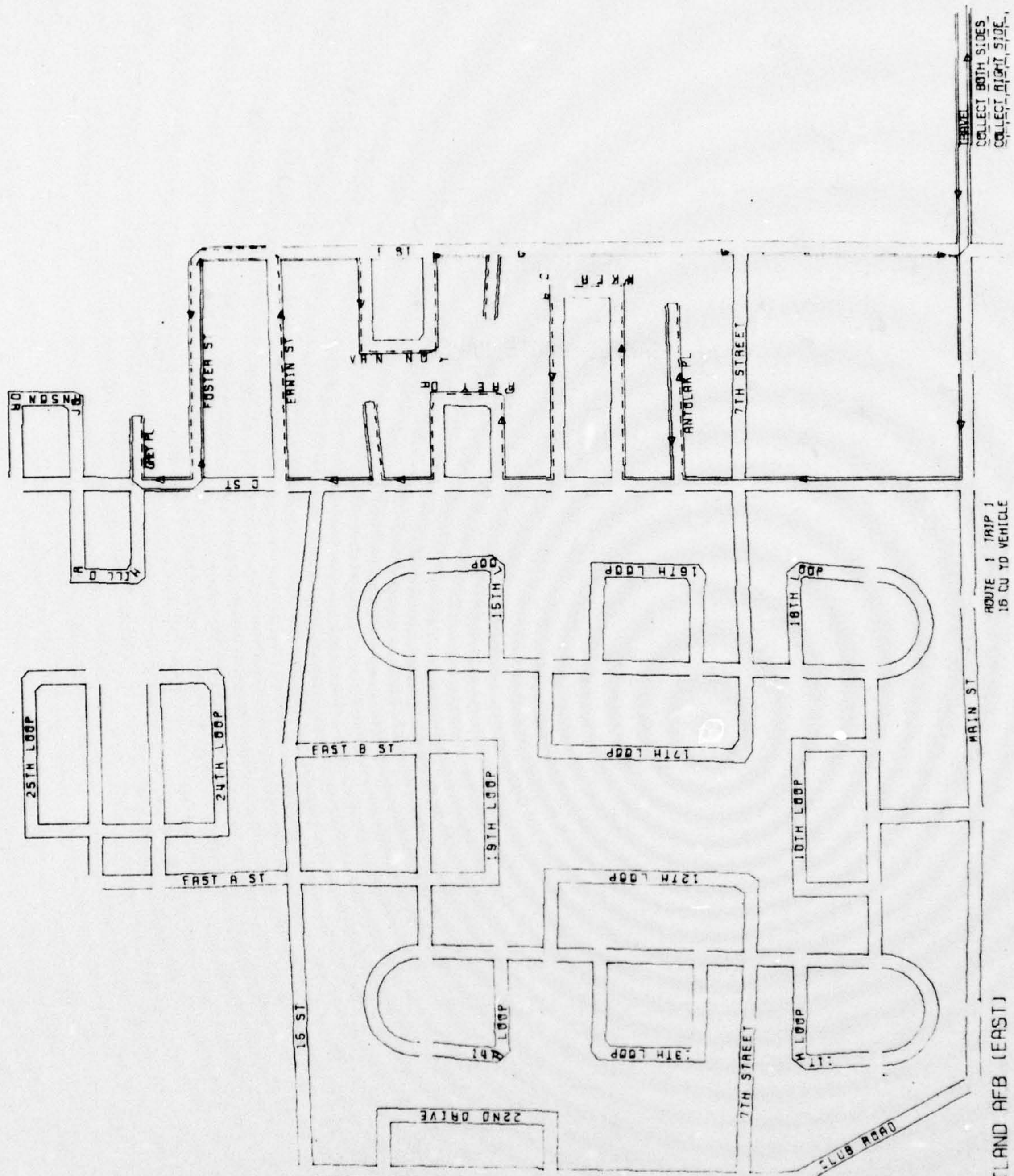
SAMPLE INPUT DATA

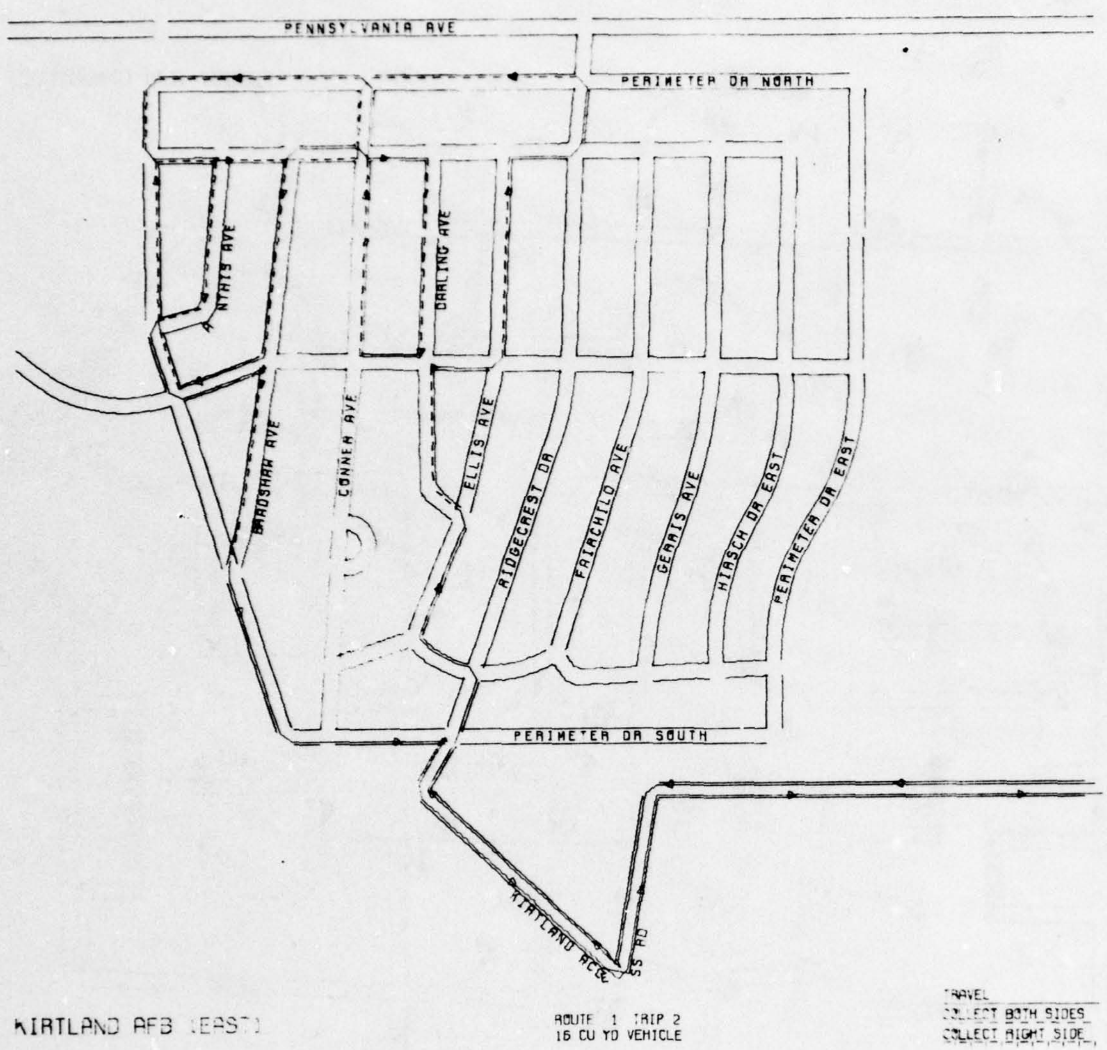
KIRTLAND AFB (EAST)
HOUSEHOLDS

1.		15.	4.	8.	8.	5.
30.	12.	15.	10.5	15.	14.5	
220.	16 CU YD VEHICLE					
7/8/9	(END OF RECORD)					
1	1 3.5	7.5	14.	.15	3.9	13.125
1	2 3.5	7.5	14.	.15	3.9	13.125
2	1 3.5	7.5	14.	.15	3.9	13.125
2	2 3.5	7.5	14.	.15	3.9	13.125
3	1 3.5	7.5	14.	.15	3.9	13.125
3	2 3.5	7.5	14.	.15	3.9	13.125
4	1 0.4	6.7	18.9	0.15	4.8	13.95
4	2 0.4	6.7	18.9	0.15	4.8	13.95
5	1 0.	5.	17.50	3.5	7.5	14.
5	2 0.	5.	17.50	3.5	7.5	14.
6	1 0.	5.	17.50	6.	10.	14.
6	2 0.	5.	17.50	6.	10.	14.
7	1 0.	5.	17.50	6.45	10.45	14.
7	2 0.	5.	17.50	6.45	10.45	14.
8	1 0.	5.	17.50	6.45	10.45	14.
8	2 0.	5.	17.50	6.45	10.45	14.
7/8/9	(END OF RECORD)					

APPENDIX E

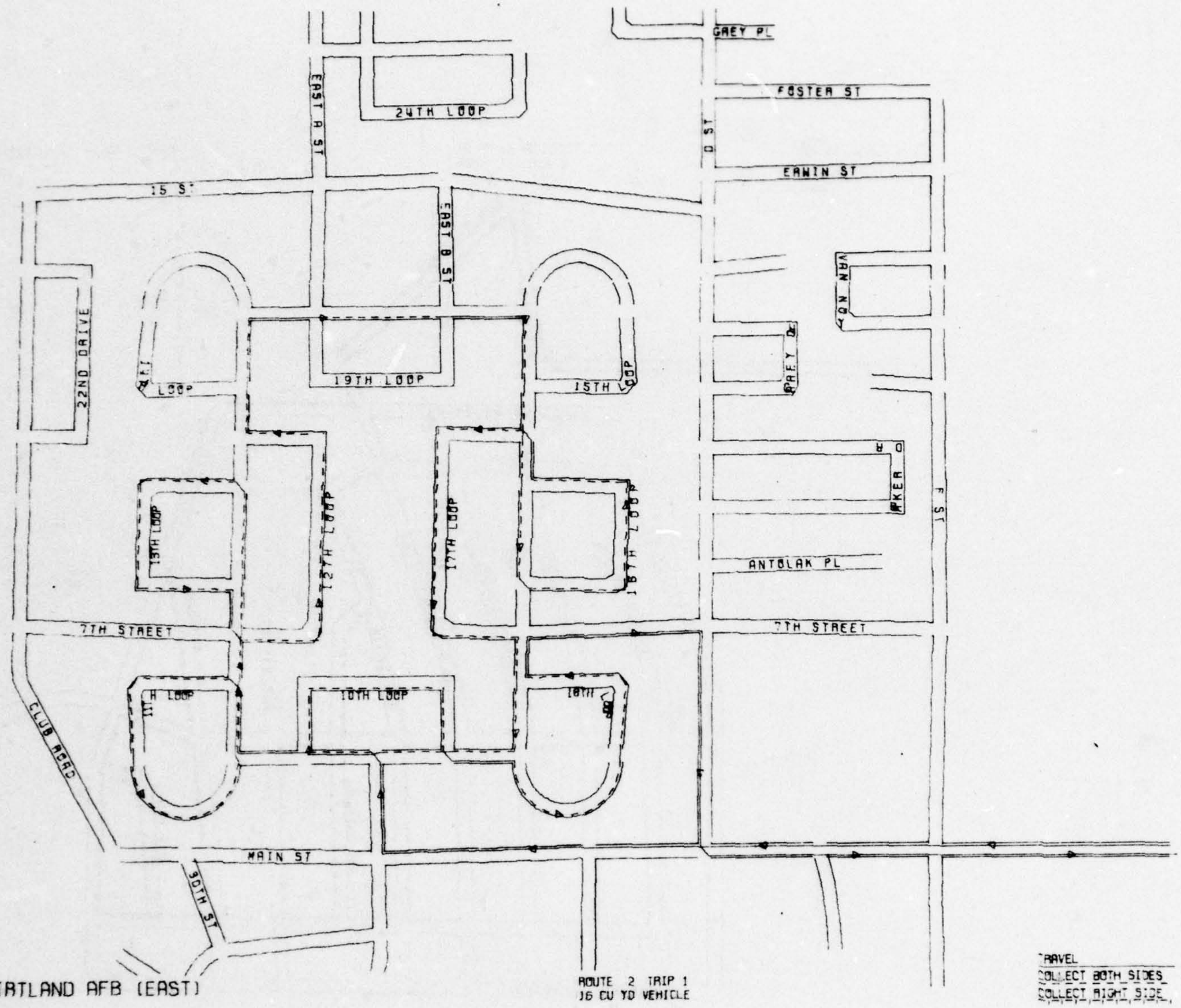
SAMPLE ROUTE MAPS


KIRTLAND AFB (EAST)
ROUTE 1 TRIP 1
16 CU YD VEHICLE
TRAVEL
COLLECT BOTH SIDES
COLLECT RIGHT SIDE
C ST
MAIN ST
7TH STREET
ANTOLAK PL
FOSTER ST
25TH LOOP
24TH LOOP
EAST A ST
EAST B ST
15 ST
15TH LOOP
16TH LOOP
17TH LOOP
18TH LOOP
19TH LOOP
10TH LOOP
12TH LOOP
22ND DRIVE
CLUB ROAD

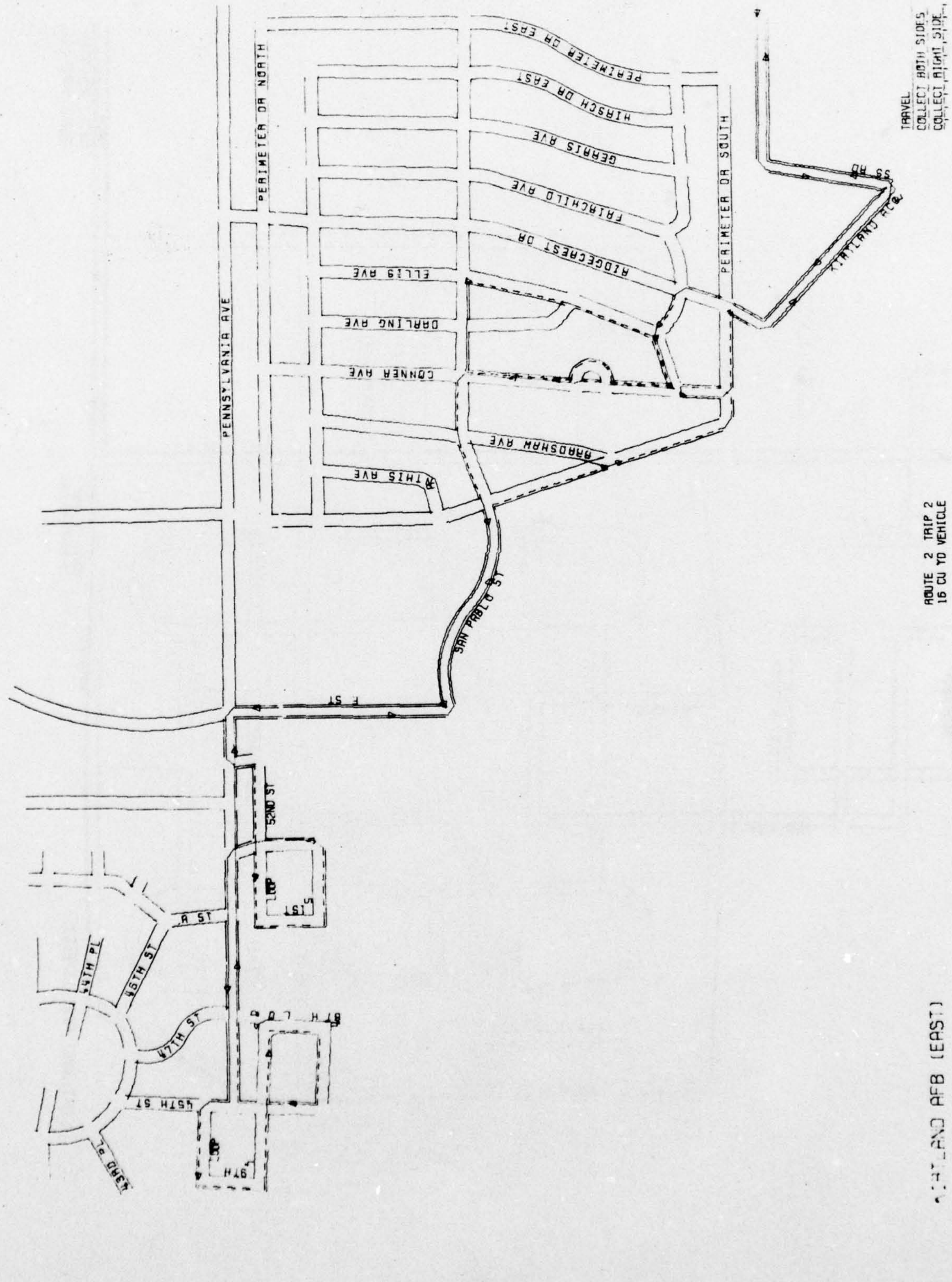


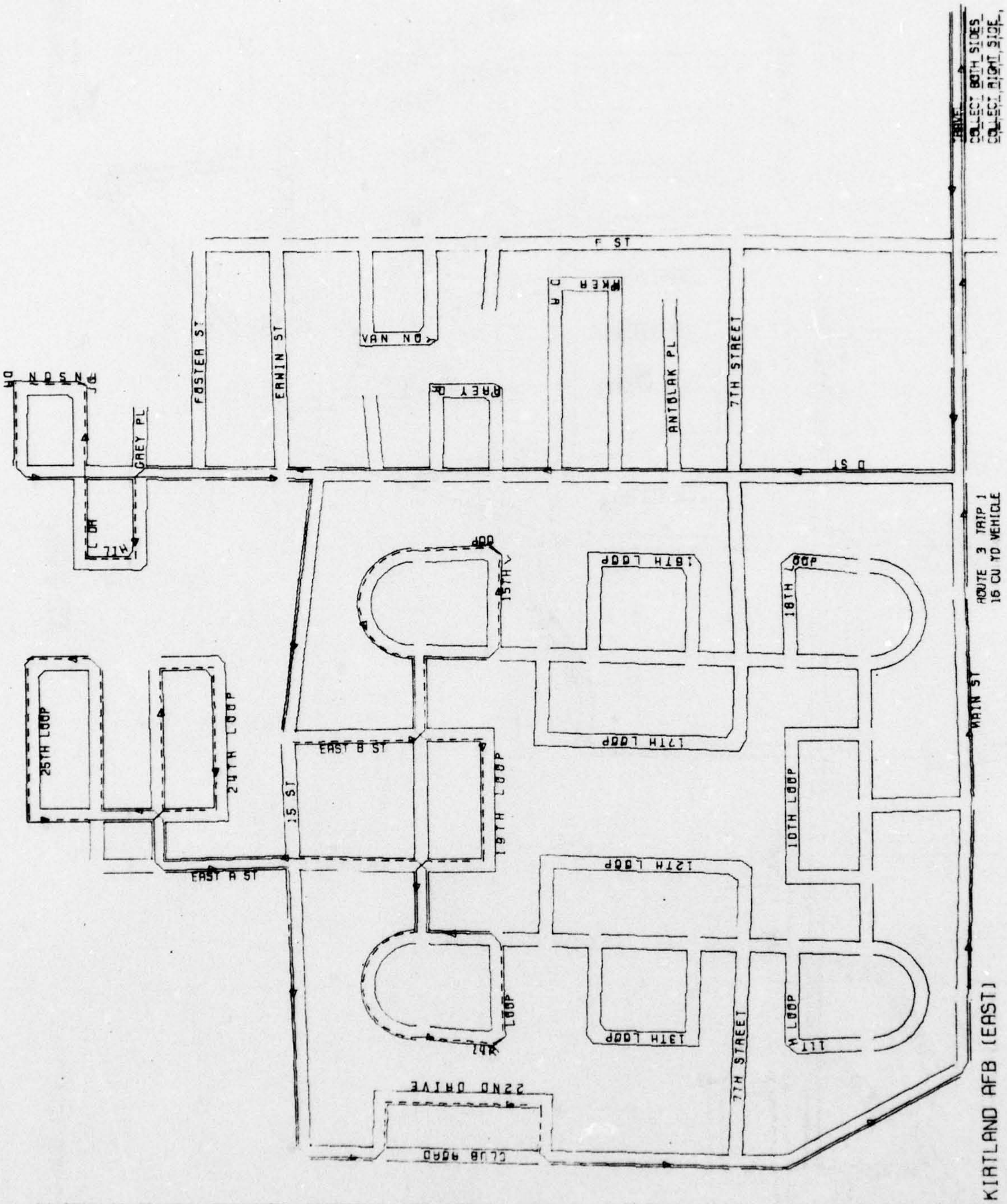
KIRTLAND AFB (EAST)

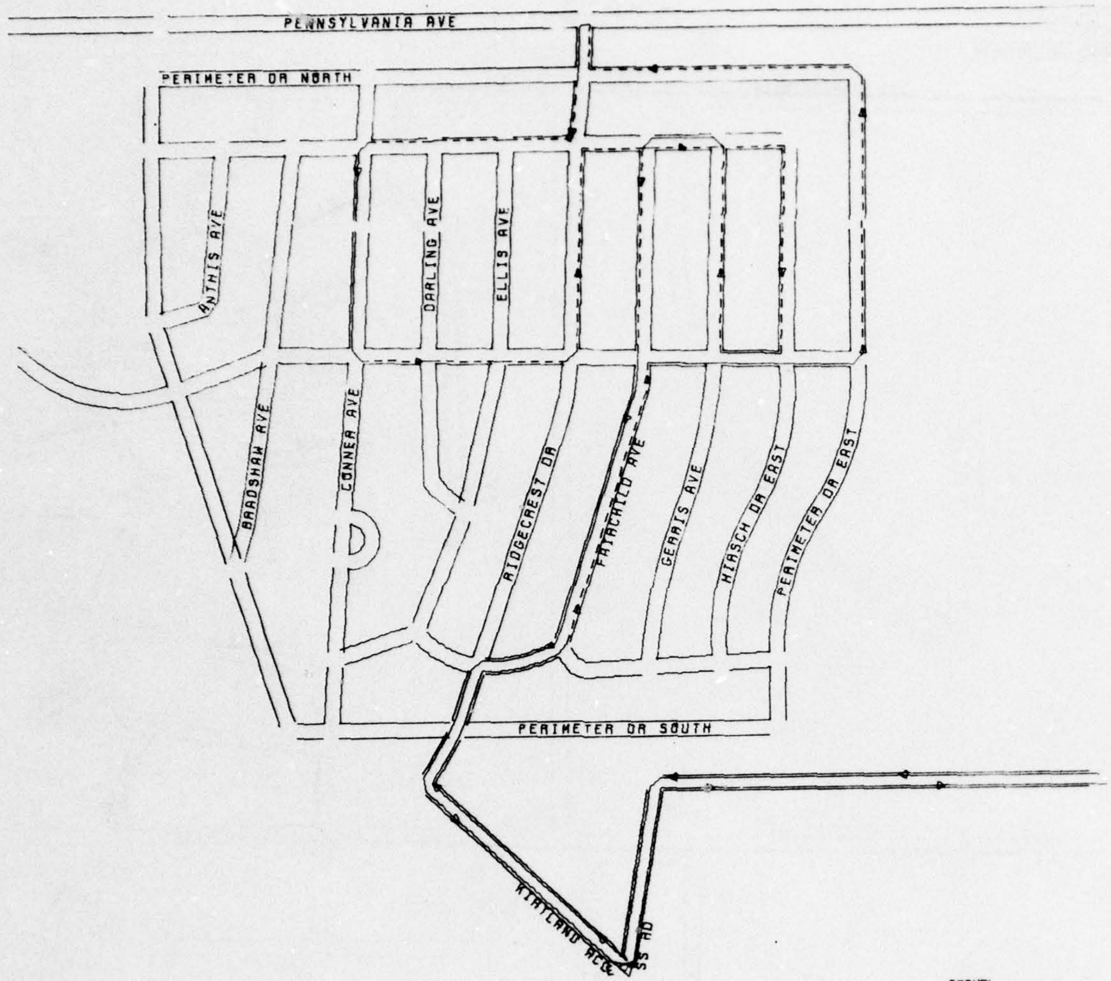
ROUTE 1 TRIP 2
15 CU YD VEHICLE



KIRTLAND AFB (EAST)


KIRTLAND AFB (EAST)
ROUTE 2 TRIP 2
16 CU YD VEHICLE
TRAVEL
COLLECT BOTH SIDES
COLLECT RIGHT SIDE
PERIMETER DR EAST
HIRSCH DR EAST
GERRIS AVE
FAIRCHILD AVE
RIDGECREST DR
ELLIS AVE
DARLING AVE
CONNER AVE
BRADSHAW AVE
ATHIS AVE
PERIMETER DR NORTH
PERIMETER DR SOUTH
PENNSYLVANIA AVE
SS RD
SAN PABLO ST
F ST
52ND ST
A ST
45TH ST
47TH ST
45TH ST

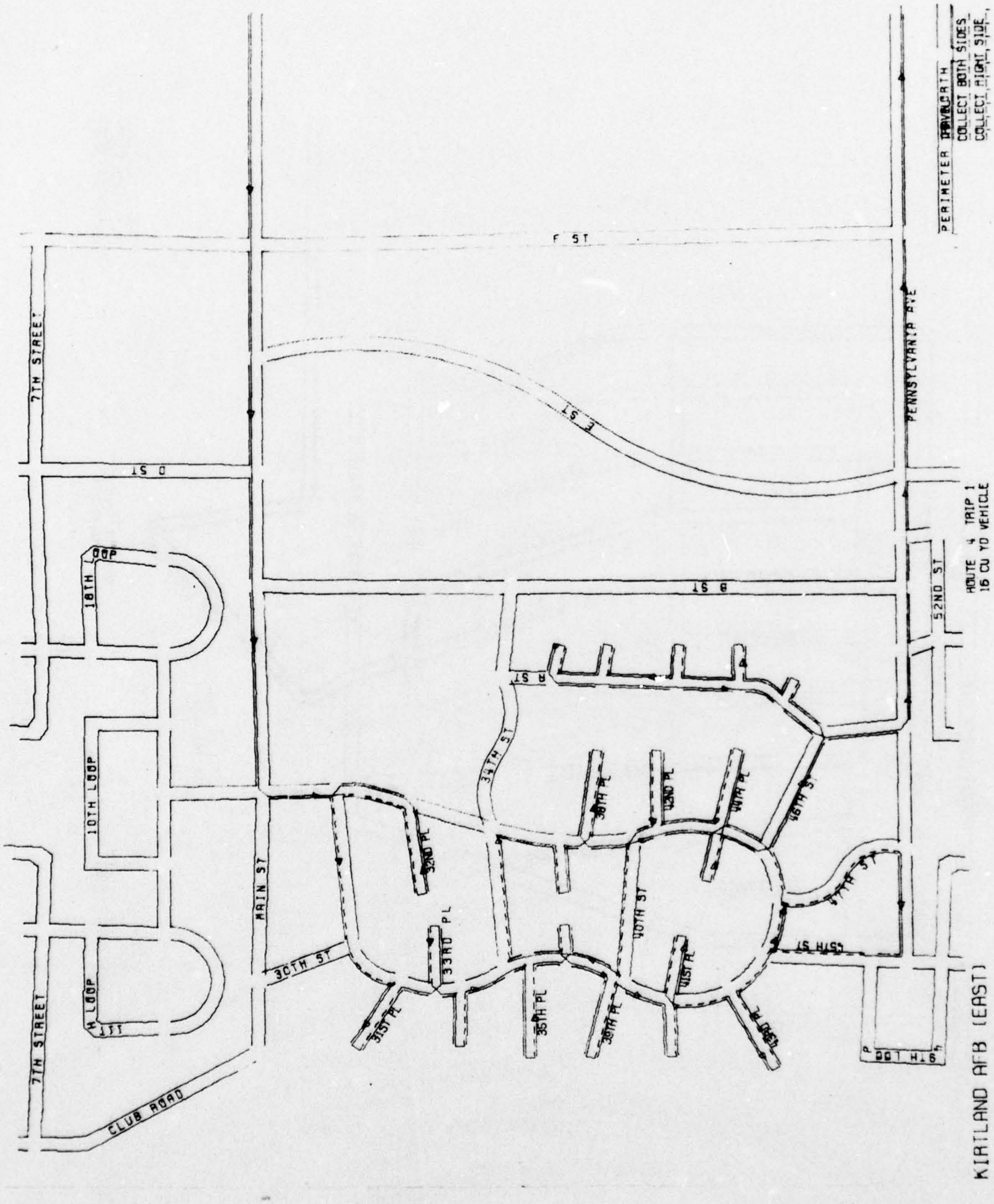

KIRTLAND AFB (EAST)
ROUTE 3 TRIP 1
16 CU YD VEHICLE
DRIVE
COLLECT BOTH SIDES
COLLECT RIGHT SIDE
F ST
D ST
MAIN ST
7TH STREET
ANTOLAK PL
FOSTER ST
ERWIN ST
GREY PL
VAN NOY
15 ST
EAST A ST
EAST B ST
25TH LOOP
24TH LOOP
19TH LOOP
16TH LOOP
17TH LOOP
18TH LOOP
10TH LOOP
12TH LOOP
13TH LOOP
22ND DRIVE
CLUB ROAD

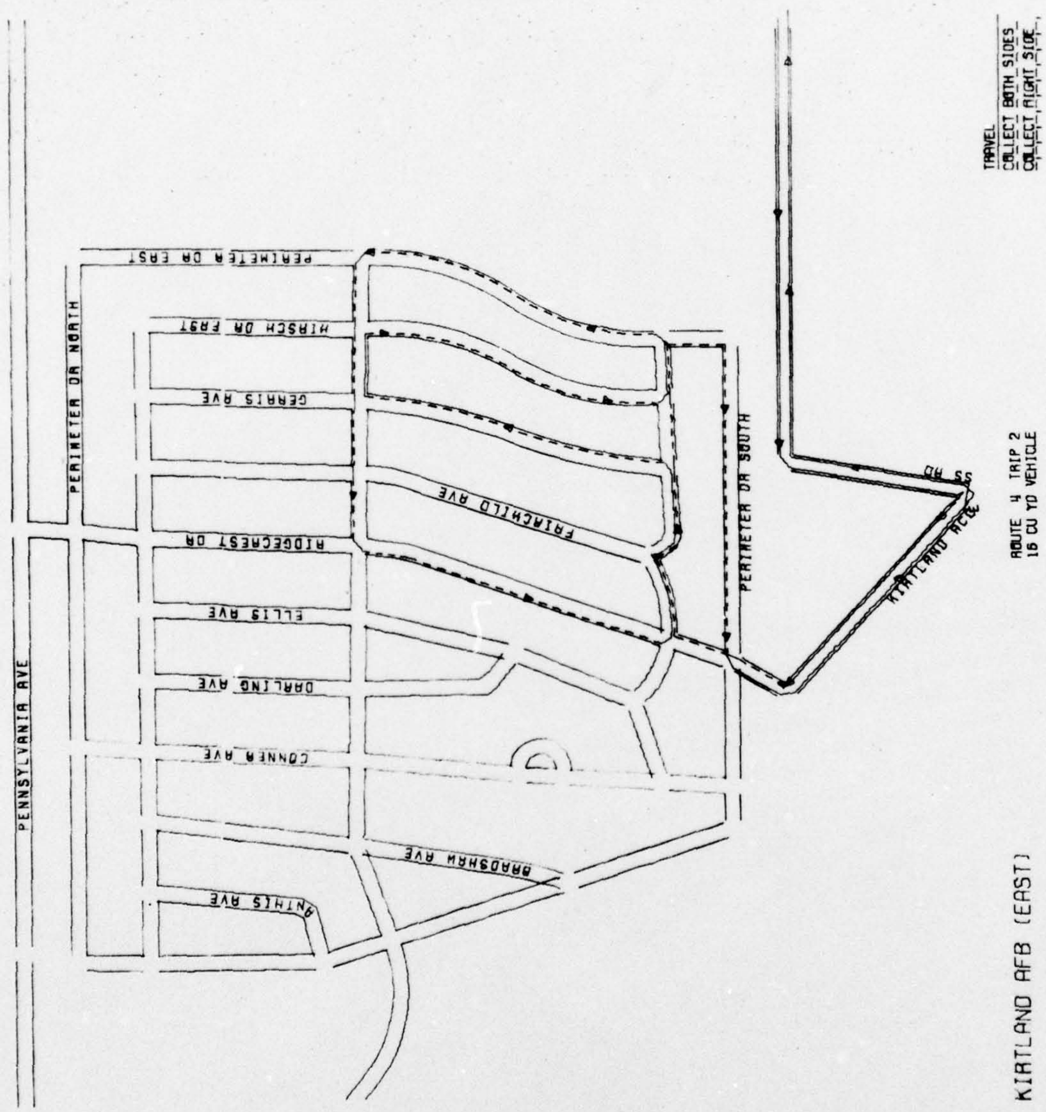


KIRTLAND AFB (EAST)

ROUTE 3 TRIP 2
16 CU YD VEHICLE

TRAVEL
COLLECT BOTH SIDES
COLLECT RIGHT SIDE


PERIMETER
COLLECT BOTH SIDES
COLLECT RIGHT SIDE
ROUTE 4 TRIP 1
16 CU YD VEHICLE
KIRTLAND AFB (EAST)
7TH STREET
F ST
E ST
D ST
B ST
A ST
PENNSYLVANIA AVE
52ND ST
18TH LOOP
10TH LOOP
MAIN ST
30TH ST
31ST PL
32ND PL
33RD PL
34TH ST
35TH PL
38TH PL
40TH ST
42ND PL
44TH PL
45TH PL
46TH ST
CLUB ROAD
9TH LOOP



(The reverse of this page is blank.)

APPENDIX F

SAMPLE PRINTED OUTPUT

```
          KIRTLAND AFB (FAST)

DATA WERE READ FOR  250 SEGMENTS AND  181 NODES

  94 STREET NUMBERS AND NAMES WERE READ FROM FILE TAPE3


INPUT DATA

UNITS OF REFUSE QUANTITY=HOUSEHOLDS

STOP TIME PER HOUSEHOLD =       1.00 MINUTES

STOP TIME PER UNIT REFUSE =     0.00 MINUTES

UNLOADING TIME =      15.00 MINUTES

MAXIMUM TRIP TIME =    4.00 HOURS

MAXIMUM ROUTE TIME =  8.00 HOURS

STARTING TIME =   8:00

VEHICLE SPEED DURING COLLECTION =      5. MPH

DURATION OF LUNCH =     30. MINUTES, STARTING AT ABOUT  12:00

DURATION OF FIRST BREAK =     15. MINUTES, STARTING AT ABOUT  10:30

DURATION OF SECOND BREAK =   15. MINUTES, STARTING AT ABOUT  14:30


VEHICLE INFORMATION

    CAPACITY         IDENTIFICATION
(HOUSEHOLDS            )

      220.00     16 CU YD VEHICLE
```

BOUNDS WERE SPECIFIED FOR 16 MAPS

SECTION	TRIP	XMIN	XMAX	XLEN	YMIN	YMAX	YLEN
1	1	3.50	7.50	14.00	.15	3.90	13.13
1	2	3.50	7.50	14.00	.15	3.90	13.13
2	1	3.50	7.50	14.00	.15	3.90	13.13
2	2	3.50	7.50	14.00	.15	3.90	13.13
3	1	3.50	7.50	14.00	.15	3.90	13.13
3	2	3.50	7.50	14.00	.15	3.90	13.13
4	1	.40	6.70	18.90	.15	4.80	13.95
4	2	.40	6.70	18.90	.15	4.80	13.95
5	1	0.00	5.00	17.50	3.50	7.50	14.00
5	2	0.00	5.00	17.50	3.50	7.50	14.00
6	1	0.00	5.00	17.50	6.00	10.00	14.00
6	2	0.00	5.00	17.50	6.00	10.00	14.00
7	1	0.00	5.00	17.50	6.45	10.45	14.00
7	2	0.00	5.00	17.50	6.45	10.45	14.00
8	1	0.00	5.00	17.50	6.45	10.45	14.00
8	2	0.00	5.00	17.50	6.45	10.45	14.00

WHERE NO BOUNDS WERE SPECIFIED, THE MAP WILL SHOW TRAVEL IN THE COLLECTION REGION BUT NOT NECESSARILY THE PATH TO OR FROM THE GARAGE OR LANDFILL.

SECTION	SCALE	XMIN	XMAX	YMIN	YMAX

SECTION	TRIP	DISTANCE (MILES)	TIME (MINUTES)	HOUSEHOLDS	CAPACITY (HOUSEHOLDS)	LOAD)
1	1	5.0	231.2	203	220.0	203.0
1	2	5.7	231.6	203	220.0	203.0
2	1	5.1	236.3	207	220.0	207.0
2	2	6.4	239.0	207	220.0	207.0
3	1	5.2	245.7	216	220.0	216.0
3	2	6.6	248.6	216	220.0	216.0
4	1	6.8	244.1	208	220.0	208.0
4	2	8.0	246.1	208	220.0	208.0
5	1	8.6	250.1	208	220.0	208.0
5	2	10.1	252.8	208	220.0	208.0
6	1	8.0	245.7	200	220.0	200.0
6	2	10.2	249.9	200	220.0	200.0
7	1	7.1	224.1	192	220.0	192.0
7	2	9.2	226.3	192	220.0	192.0
8	1	9.4	249.9	204	220.0	204.0
8	2	11.5	254.1	204	220.0	204.0

ONLY HALF OF THESE TRIPS WILL BE USED.

THE MAXIMUM TRIP TIME WILL BE EXTENDED TO 4.17 HOURS FOR VEHICLES OF CAPACITY 220.0
IF THIS IS UNSATISFACTORY, PERUN PROGRAMS PHASE2 AND PHASE3 WITH A SMALLER TIME LIMIT IN PHASE2

FINAL ROUTE SUMMARY KIRTLAND AFB (EAST)

ROUTE	VEHICLE IDENTIFICATION	VEHICLE CAPACITY (HOJSEHOLDS	SECTION(S) TRIP1	) TRIP2	DISTANCE (MILES)	TIME (HR:MIN)	HOUSEHOLDS SERVICED	REFUSE QUANTITY (HOJSEHOLDS
1	15 CU YD VEHICLE	220.0	7	3	15.8	8:28	408	408.0
2	15 CU YD VEHICLE	220.0	6	4	18.1	8:47	408	408.0
3	15 CU Y) VEHICLE	220.0	8	2	17.9	8:44	411	411.0
4	15 CU Y) VEHICLE	220.0	5	1	16.4	8:37	411	411.0
					=====	======	====	======
				TOTALS	68.3	34:36	1638	1638.0

ROUTE 1 KIRTLAND AFB (EAST) 16 CU YD VEHICLE

ACTION			SPEED (MPH)	TIME (HR:MIN)	DISTANCE (MILES)	HOUSEHOLDS SERVICED	LOAD (PCT)
LEAVE GARAGE				8:00			
DRIVE ON	D ST	TO MAIN ST	30	8:00	.2		
DRIVE ON	MAIN ST	TO D ST	25	8:02	.3		
DRIVE ON	D ST	TO ANTOLAK PL	25	8:03	.2		
PICK UP ON BOTH SIDES	ANTOLAK PL	TO	5	8:33	.2	28	12
DRIVE ON	ANTOLAK PL	TO D ST	15	8:33	.2		12
DRIVE ON	D ST	TO BAKER DR	25	8:33	.1		12
PICK UP ON BOTH SIDES	BAKER DR	TO D ST	5	9:33	.4	55	37
DRIVE ON	D ST	TO CAREY DR	25	9:33	.1		37
PICK UP ON BOTH SIDES	CAREY DR	TO D ST	5	9:56	.2	21	47
DRIVE ON	D ST	TO DUNHAM PL	25	9:56	.1		47
PICK UP ON BOTH SIDES	DUNHAM PL	TO	5	10:08	.1	11	52
DRIVE ON	DUNHAM PL	TO D ST	15	10:08	.1		52
DRIVE ON	D ST	TO ERWIN ST	25	10:08	.1		52
PICK UP ON BOTH SIDES	ERWIN ST	TO F ST	5	10:31	.2	20	61
BREAK TIME				10:31 TO 10:45			
PICK UP ON BOTH SIDES	F ST	TO FOSTER ST	5	10:51	.1	5	63
PICK UP ON BOTH SIDES	FOSTER ST	TO D ST	5	11:13	.2	20	72
DRIVE ON	D ST	TO HILL DR	25	11:14	.1		72
PICK UP ON BOTH SIDES	GREY PL	TO	5	11:26	.1	12	78
DRIVE ON	GREY PL	TO HILL DR	15	11:27	.1		78
DRIVE ON	D ST	TO FOSTER ST	25	11:27	.1		78
DRIVE ON	FOSTER ST	TO F ST	15	11:27	.2		78
DRIVE ON	F ST	TO VAN NOY	15	11:28	.1		78
PICK UP ON BOTH SIDES	VAN NOY	TO F ST	5	11:43	.2	13	84
DRIVE ON	F ST	TO WARD PLACE	15	11:43	.1		84
PICK UP ON BOTH SIDES	WARD PLACE	TO	5	11:51	.1	7	87
DRIVE ON	WARD PLACE	TO F ST	15	11:51	.1		87
DRIVE ON	F ST	TO MAIN ST	15	11:53	.4		87
DRIVE ON	MAIN ST	TO D ST	25	11:55	.7		87
DRIVE ON	D ST	TO KIRTLAND ACCESS RD	30	11:57	1.1		87
DRIVE ON	KIRTLAND ACCESS RD	TO WEST ORDINANCE RD	30	11:57	.3		87
DRIVE ON	WEST ORDINANCE RD	TO LAND FILL	30	11:59	1.3		87
UNLOAD				11:59 TO 12:14			
BREAK FOR LUNCH				12:14 TO 12:44			
LEAVE LAND FILL				12:44			
DRIVE ON	WEST ORDINANCE RD	TO KIRTLAND ACCESS RD	30	12:46	1.3		
DRIVE ON	KIRTLAND ACCESS RD	TO RIDGECREST DR	30	12:48	.9		
DRIVE ON	RIDGECREST DR	TO FAIRCHILD AVE	15	12:48	.1		
DRIVE ON	FAIRCHILD AVE	TO ELLIS AVE	15	12:48	.1		
DRIVE ON	ELLIS AVE	TO DARLING AVE	15	12:49	.1		
PICK UP ON BOTH SIDES	DARLING AVE	TO SAN PABLO ST	5	13:04	.1	14	6
DRIVE ON	SAN PABLO ST	TO ELLIS AVE	15	13:04	.1		6
PICK UP ON BOTH SIDES	ELLIS AVE	TO HIRSCH DR NORTH	5	13:25	.2	19	14
DRIVE ON	HIRSCH DR NORTH	TO RIDGECREST DR	15	13:25	.1		14
DRIVE ON	RIDGECREST DR	TO PERIMETER DR NORTH	15	13:26	.1		14
PICK UP ON BOTH SIDES	PERIMETER DR NORTH	TO CONNER AVE	5	13:46	.2	19	23
PICK UP ON BOTH SIDES	CONNER AVE	TO HIRSCH DR NORTH	5	13:50	.1	3	25

ACTION			SPEED (MPH)	TIME (HR:MIN)	DISTANCE (MILES)	HOUSEHOLDS SERVICED	LOAD (PCT)
DRIVE ON	HIRSCH DR NORTH	TO DARLING AVE	15	13:50	.1		25
PICK UP ON BOTH SIDES	DARLING AVE	TO SAN PABLO ST	5	14:10	.2	18	33
DRIVE ON	SAN PABLO ST	TO CONNER AVE	15	14:10	.1		33
PICK UP ON BOTH SIDES	CONNER AVE	TO HIRSCH DR NORTH	5	14:31	.2	19	41
BREAK TIME				14:31 TO 14:45			
DRIVE ON	CONNER AVE	TO PERIMETER DR NORTH	15	14:46	.1		41
PICK UP ON BOTH SIDES	PERIMETER DR NORTH	TO PERIMETER DR WEST	5	15:08	.2	20	50
PICK UP ON BOTH SIDES	PERIMETER DR WEST	TO HIRSCH DR NORTH	5	15:15	.1	6	53
PICK UP ON BOTH SIDES	HIRSCH DR NORTH	TO ANTHIS AVE	5	15:20	.1	5	55
PICK UP ON BOTH SIDES	HIRSCH DR NORTH	TO BRADSHAW AVE	5	15:27	.1	6	58
PICK UP ON BOTH SIDES	HIRSCH DR NORTH	TO CONNER AVE	5	15:32	.1	5	60
DRIVE ON	HIRSCH DR NORTH	TO BRADSHAW AVE	15	15:33	.1		60
PICK UP ON BOTH SIDES	BRADSHAW AVE	TO SAN PABLO ST	5	15:54	.2	20	69
DRIVE ON	SAN PABLO ST	TO PERIMETER DR WEST	15	15:55	.1		69
PICK UP ON BOTH SIDES	PERIMETER DR WEST	TO ANTHIS AVE	5	16:00	.1	5	72
PICK UP ON BOTH SIDES	PERIMETER DR WEST	TO HIRSCH DR NORTH	5	16:19	.1	17	79
DRIVE ON	HIRSCH DR NORTH	TO ANTHIS AVE	15	16:19	.1		79
PICK UP ON BOTH SIDES	ANTHIS AVE	TO PERIMETER DR WEST	5	16:41	.2	20	89
DRIVE ON	PERIMETER DR WEST	TO SAN PABLO ST	15	16:41	.1		89
DRIVE ON	SAN PABLO ST	TO BRADSHAW AVE	15	16:41	.1		89
PICK UP ON BOTH SIDES	BRADSHAW AVE	TO PERIMETER DR WEST	5	17:03	.2	20	98
DRIVE ON	PERIMETER DR WEST	TO PERIMETER DR SOUTH	15	17:04	.1		98
DRIVE ON	PERIMETER DR SOUTH	TO RIDGECREST DR	15	17:04	.1		98
DRIVE ON	RIDGECREST DR	TO KIRTLAND ACCESS RD	15	17:04	.0		98
DRIVE ON	KIRTLAND ACCESS RD	TO WEST ORDINANCE RD	30	17:06	.3		98
DRIVE ON	WEST ORDINANCE RD	TO LAND FILL	30	17:08	1.0		98
UNLOAD				17:08 TO 17:23			
DRIVE ON	WEST ORDINANCE RD	TO KIRTLAND ACCESS RD	30	17:25	1.0		
DRIVE ON	KIRTLAND ACCESS RD	TO O ST	30	17:25	.3		
DRIVE ON	O ST	TO GARAGE	30	17:27	.3		

ROUTE 2 KIRTLAND AFB (EAST) 16 CU YD VEHICLE

ACTION			SPEED (MPH)	TIME (HR:MIN)	DISTANCE (MILES)	HOUSEHOLDS SERVICED	LOAD (PCT)
LEAVE GARAGE				8:00			
DRIVE ON	O ST	TO MAIN ST	30	8:00	.2		
DRIVE ON	MAIN ST	TO WEST SANDIA DR (S)	25	8:03	1.1		
DRIVE ON	WEST SANDIA DR (S)	TO EAST SANDIA DRIVE (WEST)	15	8:03	.1		
PICK UP ON BOTH SIDES	EAST SANDIA DRIVE (WEST)	TO 10TH LOOP	5	8:07	.1	3	1
DRIVE ON	EAST SANDIA DRIVE (WEST)	TO 11TH LOOP	15	8:07	.1		1
PICK UP ON BOTH SIDES	EAST SANDIA DRIVE (NORTH)	TO 11TH LOOP	5	8:14	.1	6	4
PICK UP ON BOTH SIDES	11TH LOOP	TO	5	8:29	.1	13	9
PICK UP ON BOTH SIDES	11TH LOOP	TO EAST SANDIA DRIVE (WEST)	5	8:42	.1	12	15
DRIVE ON	EAST SANDIA DRIVE (NORTH)	TO 11TH LOOP	15	8:42	.1		15
PICK UP ON BOTH SIDES	EAST SANDIA DRIVE (NORTH)	TO 7TH STREET	5	8:45	.0	2	16
PICK UP ON	EAST SANDIA DRIVE (NORTH)	TO 13TH LOOP	5	8:46	.0	1	16
PICK UP ON BOTH SIDES	EAST SANDIA DRIVE (NORTH)	TO 13TH LOOP	5	8:51	.1	4	18
PICK UP ON BOTH SIDES	13TH LOOP	TO EAST SANDIA DRIVE (NORTH)	5	9:13	.2	19	27
DRIVE ON	EAST SANDIA DRIVE (NORTH)	TO 7TH STREET	15	9:13	.0		27
PICK UP ON BOTH SIDES	12TH LOOP	TO EAST SANDIA DRIVE (NORTH)	5	9:41	.3	24	38
PICK UP ON	EAST SANDIA DRIVE (NORTH)	TO 14TH LOOP	5	9:42	.0	1	38
PICK UP ON BOTH SIDES	EAST SANDIA DRIVE (NORTH)	TO 14TH LOOP	5	9:45	.1	2	39
DRIVE ON	EAST SANDIA DR (EAST)	TO 13TH LOOP	15	9:45	.1		39
PICK UP ON BOTH SIDES	EAST SANDIA DR (EAST)	TO 19TH LOOP	5	9:51	.1	5	41
DRIVE ON	EAST SANDIA DR (EAST)	TO 15TH LOOP	15	9:52	.1		41
PICK UP ON BOTH SIDES	EAST SANDIA DRIVE (SOUTH)	TO 15TH LOOP	5	9:55	.1	3	43
PICK UP ON BOTH SIDES	EAST SANDIA DRIVE (SOUTH)	TO 17TH LOOP	5	9:58	.0	2	44
PICK UP ON	EAST SANDIA DRIVE (SOUTH)	TO 16TH LOOP	5	9:59	.0	1	44
PICK UP ON BOTH SIDES	EAST SANDIA DRIVE (SOUTH)	TO 16TH LOOP	5	10:07	.1	7	47
PICK UP ON BOTH SIDES	16TH LOOP	TO EAST SANDIA DRIVE (SOUTH)	5	10:28	.2	18	55
DRIVE ON	EAST SANDIA DRIVE (SOUTH)	TO 17TH LOOP	15	10:28	.0		55
PICK UP ON BOTH SIDES	17TH LOOP	TO 7TH STREET	5	10:56	.3	24	66
BREAK TIME				10:56 TO 11:11			
PICK UP ON	EAST SANDIA DRIVE (SOUTH)	TO 18TH LOOP	5	11:12	.0	1	67
PICK UP ON BOTH SIDES	EAST SANDIA DRIVE (SOUTH)	TO 18TH LOOP	5	11:17	.1	4	69
PICK UP ON BOTH SIDES	EAST SANDIA DRIVE (WEST)	TO 10TH LOOP	5	11:20	.1	2	69
PICK UP ON	EAST SANDIA DRIVE (WEST)	TO WEST SANDIA DR (S)	5	11:21	.1	1	70
DRIVE ON	EAST SANDIA DRIVE (WEST)	TO 10TH LOOP	15	11:22	.1		70
PICK UP ON BOTH SIDES	10TH LOOP	TO EAST SANDIA DRIVE (WEST)	5	11:42	.2	18	78
DRIVE ON	EAST SANDIA DRIVE (WEST)	TO 18TH LOOP	15	11:43	.1		78
PICK UP ON BOTH SIDES	18TH LOOP	TO	5	11:56	.1	12	84
PICK UP ON BOTH SIDES	18TH LOOP	TO EAST SANDIA DRIVE (SOUTH)	5	12:13	.1	15	90
BREAK FOR LUNCH				12:13 TO 12:43			
DRIVE ON	EAST SANDIA DRIVE (SOUTH)	TO 7TH STREET	15	12:43	.0		90
DRIVE ON	7TH STREET	TO D ST	15	12:44	.2		90
DRIVE ON	D ST	TO MAIN ST	25	12:44	.2		90
DRIVE ON	MAIN ST	TO O ST	25	12:46	.9		90
DRIVE ON	O ST	TO KIRTLAND ACCESS RD	30	12:48	1.1		90
DRIVE ON	KIRTLAND ACCESS RD	TO WEST ORDINANCE RD	30	12:49	.3		90
DRIVE ON	WEST ORDINANCE RD	TO LAND FILL	30	12:51	1.0		90
UNLOAD				12:51 TO 13:05			
LEAVE LAND FILL				13:06			

ACTION			SPEED (MPH)	TIME (HR:MIN)	DISTANCE (MILES)	HOUSEHOLDS SERVICED	LOAD (PCT)
DRIVE ON	WEST ORDINANCE RD	TO KIRTLAND ACCESS RD	30	13:08	1.0		
DRIVE ON	KIRTLAND ACCESS RD	TO RIDGECREST DR	30	13:09	.9		
DRIVE ON	RIDGECREST DR	TO FAIRCHILD AVE	15	13:10	.1		
PICK UP ON BOTH SIDES	FAIRCHILD AVE	TO ELLIS AVE	5	13:14	.1	4	1
PICK UP ON BOTH SIDES	ELLIS AVE	TO CONNER AVE	5	13:20	.1	5	4
PICK UP ON BOTH SIDES	CONNER AVE	TO	5	13:31	.1	10	8
PICK UP ON BOTH SIDES	CONNER AVE	TO	5	13:37	.1	5	10
PICK UP ON BOTH SIDES	CONNER AVE	TO SAN PABLO ST	5	13:53	.1	15	17
DRIVE ON	SAN PABLO ST	TO ELLIS AVE	15	13:53	.1		17
PICK UP ON BOTH SIDES	ELLIS AVE	TO DARLING AVE	5	14:09	.1	14	24
PICK UP ON BOTH SIDES	ELLIS AVE	TO FAIRCHILD AVE	5	14:24	.1	14	30
DRIVE ON	ELLIS AVE	TO CONNER AVE	15	14:24	.1		30
DRIVE ON	CONNER AVE	TO	15	14:24	.1		30
PICK UP ON BOTH SIDES	CONNER AVE	TO	5	14:27	.0	2	31
DRIVE ON	CONNER AVE	TO SAN PABLO ST	15	14:27	.1		31
PICK UP ON BOTH SIDES	SAN PABLO ST	TO BRADSHAW AVE	5	14:32	.1	4	33
BREAK TIME				14:32 TO 14:47			
PICK UP ON BOTH SIDES	SAN PABLO ST	TO PERIMETER DR WEST	5	14:56	.1	8	36
DRIVE ON	SAN PABLO ST	TO E ST	15	14:57	.2		36
DRIVE ON	E ST	TO PENNSYLVANIA AVE	35	14:57	.2		36
DRIVE ON	PENNSYLVANIA AVE	TO 53RD ST	25	14:57	.1		36
DRIVE ON	53RD ST	TO 52ND ST	15	14:57	.0		36
PICK UP ON BOTH SIDES	52ND ST	TO 51ST LOOP	5	15:06	.1	8	40
PICK UP ON BOTH SIDES	51ST LOOP	TO 50TH ST	5	15:29	.2	20	49
DRIVE ON	50TH ST	TO PENNSYLVANIA AVE	15	15:29	.1		49
DRIVE ON	PENNSYLVANIA AVE	TO 45TH ST	25	15:30	.3		49
DRIVE ON	45TH ST	TO 49TH LOOP	15	15:30	.0		49
PICK UP ON BOTH SIDES	49TH LOOP	TO 48TH LOOP	5	15:55	.2	22	59
PICK UP ON BOTH SIDES	48TH LOOP	TO 45TH ST	5	16:19	.2	22	69
DRIVE ON	45TH ST	TO PENNSYLVANIA AVE	15	16:20	.1		69
PICK UP ON BOTH SIDES	PENNSYLVANIA AVE	TO 47TH ST	5	16:25	.1	4	71
DRIVE ON	PENNSYLVANIA AVE	TO E ST	25	16:25	.3		71
DRIVE ON	E ST	TO SAN PABLO ST	35	16:26	.2		71
DRIVE ON	SAN PABLO ST	TO PERIMETER DR WEST	15	16:27	.2		71
PICK UP ON BOTH SIDES	PERIMETER DR WEST	TO BRADSHAW AVE	5	16:45	.1	17	79
PICK UP ON BOTH SIDES	PERIMETER DR WEST	TO PERIMETER DR SOUTH	5	17:03	.1	16	86
PICK UP ON BOTH SIDES	PERIMETER DR SOUTH	TO CONNER AVE	5	17:06	.0	3	87
PICK UP ON BOTH SIDES	CONNER AVE	TO ELLIS AVE	5	17:12	.1	5	89
DRIVE ON	CONNER AVE	TO PERIMETER DR SOUTH	15	17:12	.1		89
PICK UP ON BOTH SIDES	PERIMETER DR SOUTH	TO RIDGECREST DR	5	17:23	.1	10	94
DRIVE ON	RIDGECREST DR	TO KIRTLAND ACCESS RD	15	17:23	.0		94
DRIVE ON	KIRTLAND ACCESS RD	TO WEST ORDINANCE RD	30	17:25	.9		94
DRIVE ON	WEST ORDINANCE RD	TO LAND FILL	30	17:27	1.0		94
UNLOAD				17:27 TO 17:42			
DRIVE ON	WEST ORDINANCE RD	TO KIRTLAND ACCESS RD	30	17:44	1.0		
DRIVE ON	KIRTLAND ACCESS RD	TO D ST	30	17:44	.3		
DRIVE ON	D ST	TO GARAGE	30	17:46	.9		

ROUTE 3 KIRTLAND AFB (EAST) 16 CU YD VEHICLE

ACTION			SPEED (MPH)	TIME (HR:MIN)	DISTANCE (MILES)	HOUSEHOLDS SERVICED	LOAD (PCT)
LEAVE GARAGE				8:00			
DRIVE ON	D ST	TO MAIN ST	30	8:00	.2		
DRIVE ON	MAIN ST	TO D ST	25	8:02	.3		
DRIVE ON	D ST	TO HILL DR	25	8:04	.7		
PICK UP ON BOTH SIDES	HILL DR	TO JOHNSON DR	5	8:32	.2	26	11
PICK UP ON BOTH SIDES	JOHNSON DR	TO D ST	5	9:04	.2	30	25
DRIVE ON	D ST	TO 15 ST	25	9:05	.3		25
DRIVE ON	15 ST	TO EAST B ST	15	9:06	.2		25
PICK UP ON	EAST B ST	TO 19TH LOOP	5	9:08	.1	1	25
PICK UP ON BOTH SIDES	EAST SANDIA DR (EAST)	TO 15TH LOOP	5	9:11	.1	2	26
DRIVE ON	EAST SANDIA DRIVE (SOUTH)	TO 15TH LOOP	15	9:11	.1		26
PICK UP ON BOTH SIDES	15TH LOOP	TO	5	9:26	.1	13	32
PICK UP ON BOTH SIDES	15TH LOOP	TO EAST SANDIA DR (EAST)	5	9:39	.1	12	38
DRIVE ON	EAST SANDIA DR (EAST)	TO 19TH LOOP	15	9:40	.1		38
PICK UP ON BOTH SIDES	19TH LOOP	TO EAST A ST	5	9:59	.2	17	45
DRIVE ON	EAST SANDIA DR (EAST)	TO 14TH LOOP	15	9:59	.1		45
PICK UP ON BOTH SIDES	14TH LOOP	TO	5	10:13	.1	12	51
PICK UP ON BOTH SIDES	14TH LOOP	TO EAST SANDIA DRIVE (NORTH)	5	10:28	.1	13	57
DRIVE ON	EAST SANDIA DRIVE (NORTH)	TO 14TH LOOP	15	10:28	.1		57
DRIVE ON	EAST SANDIA DR (EAST)	TO 19TH LOOP	15	10:28	.1		57
PICK UP ON	EAST A ST	TO 15 ST	5	10:30	.1	1	57
BREAK TIME				10:30 TO 10:45			
DRIVE ON	EAST A ST	TO 24TH LOOP	15	10:46	.1		57
DRIVE ON	24TH LOOP	TO	15	10:46	.0		57
DRIVE ON		TO 25TH LOOP	15	10:46	.1		57
PICK UP ON BOTH SIDES	25TH LOOP	TO	5	11:00	.1	13	63
PICK UP ON BOTH SIDES	25TH LOOP	TO	5	11:18	.2	15	70
DRIVE ON		TO 24TH LOOP	15	11:18	.1		70
PICK UP ON BOTH SIDES	24TH LOOP	TO	5	11:34	.1	14	76
PICK UP ON BOTH SIDES	24TH LOOP	TO	5	11:59	.2	23	87
DRIVE ON	24TH LOOP	TO EAST A ST	15	11:59	.0		87
DRIVE ON	EAST A ST	TO 15 ST	15	12:00	.1		87
DRIVE ON	15 ST	TO CLUB ROAD	15	12:01	.2		87
BREAK FOR LUNCH				12:01 TO 12:31			
DRIVE ON	CLUB ROAD	TO 22ND DRIVE	15	12:31	.1		87
PICK UP ON BOTH SIDES	22ND DRIVE	TO CLUB ROAD	5	12:46	.2	12	92
DRIVE ON	CLUB ROAD	TO MAIN ST	15	12:47	.4		92
DRIVE ON	MAIN ST	TO D ST	25	12:50	1.3		92
DRIVE ON	D ST	TO KIRTLAND ACCESS RD	30	12:52	1.1		92
DRIVE ON	KIRTLAND ACCESS RD	TO WEST ORDINANCE RD	30	12:53	.3		92
DRIVE ON	WEST ORDINANCE RD	TO LAND FILL	30	12:55	1.0		92
UNLOAD				12:55 TO 13:10			
LEAVE LAND FILL				13:10			
DRIVE ON	WEST ORDINANCE RD	TO KIRTLAND ACCESS RD	30	13:12	1.0		
DRIVE ON	KIRTLAND ACCESS RD	TO RIDGECREST DR	30	13:14	.3		
DRIVE ON	RIDGECREST DR	TO FAIRCHILD AVE	15	13:14	.1		
DRIVE ON	FAIRCHILD AVE	TO WALKER AVE	15	13:14	.1		
PICK UP ON BOTH SIDES	FAIRCHILD AVE	TO SAN PABLO ST	5	13:47	.2	30	13

ACTION				SPEED (MPH)	TIME (HR:MIN)	DISTANCE (MILES)	HOUSEHOLDS SERVICED	LOAD (PCT)
DRIVE ON		SAN PABLO ST	TO PERIMETER DR EAST	15	13:47	.2		13
PICK UP ON	BOTH SIDES	PERIMETER DR EAST	TO PERIMETER DR NORTH	5	14:18	.2	28	26
PICK UP ON	BOTH SIDES	PERIMETER DR NORTH	TO RIDGECREST DR	5	14:44	.2	24	37
BREAK TIME					14:44 TO 14:53			
PICK UP ON	BOTH SIDES	RIDGECREST DR	TO PENNSYLVANIA AVE	5	15:01	.0	2	38
DRIVE ON		RIDGECREST DR	TO PERIMETER DR NORTH	15	15:02	.0		38
PICK UP ON	BOTH SIDES	RIDGECREST DR	TO HIRSCH DR NORTH	5	15:06	.1	4	39
PICK UP ON	BOTH SIDES	HIRSCH DR NORTH	TO ELLIS AVE	5	15:12	.1	5	42
PICK UP ON	BOTH SIDES	HIRSCH DR NORTH	TO DARLING AVE	5	15:18	.1	6	44
PICK UP ON	BOTH SIDES	HIRSCH DR NORTH	TO CONNER AVE	5	15:23	.1	4	46
DRIVE ON		CONNER AVE	TO SAN PABLO ST	15	15:24	.2		46
PICK UP ON	BOTH SIDES	SAN PABLO ST	TO DARLING AVE	5	15:28	.1	4	48
PICK UP ON	BOTH SIDES	SAN PABLO ST	TO ELLIS AVE	5	15:33	.1	4	50
PICK UP ON	BOTH SIDES	SAN PABLO ST	TO RIDGECREST DR	5	15:39	.1	6	53
PICK UP ON	BOTH SIDES	RIDGECREST DR	TO HIRSCH DR NORTH	5	16:00	.2	19	61
PICK UP ON	BOTH SIDES	HIRSCH DR NORTH	TO FAIRCHILD AVE	5	16:06	.1	5	64
PICK UP ON	BOTH SIDES	HIRSCH DR NORTH	TO GERRIS AVE	5	16:11	.1	5	66
PICK UP ON	BOTH SIDES	HIRSCH DR NORTH	TO HIRSCH DR EAST	5	16:17	.1	5	68
PICK UP ON	BOTH SIDES	HIRSCH DR EAST	TO SAN PABLO ST	5	16:39	.2	20	77
DRIVE ON		SAN PABLO ST	TO GERRIS AVE	15	16:39	.1		77
PICK UP ON	BOTH SIDES	GERRIS AVE	TO HIRSCH DR NORTH	5	16:59	.2	18	85
DRIVE ON		HIRSCH DR NORTH	TO FAIRCHILD AVE	15	16:59	.1		85
PICK UP ON	BOTH SIDES	FAIRCHILD AVE	TO SAN PABLO ST	5	17:19	.2	18	94
DRIVE ON		FAIRCHILD AVE	TO RIDGECREST DR	15	17:20	.3		94
DRIVE ON		RIDGECREST DR	TO KIRTLAND ACCESS RD	15	17:20	.1		94
DRIVE ON		KIRTLAND ACCESS RD	TO WEST ORDINANCE RD	30	17:22	.9		94
DRIVE ON		WEST ORDINANCE RD	TO LAND FILL	30	17:24	1.0		94
UNLOAD					17:24 TO 17:39			
DRIVE ON		WEST ORDINANCE RD	TO KIRTLAND ACCESS RD	30	17:41	1.0		
DRIVE ON		KIRTLAND ACCESS RD	TO O ST	30	17:41	.3		
DRIVE ON		O ST	TO GARAGE	30	17:43	.9		

ROUTE - KIRTLAND AFB (EAST) 16 CU YD VEHICLE

ACTION			SPEED (MPH)	TIME (HR:MIN)	DISTANCE (MILES)	HOUSEHOLDS SERVICED	LOAD (PCT)
LEAVE GARAGE				8:00			
DRIVE ON	O ST	TO MAIN ST	30	8:00	.2		
DRIVE ON	MAIN ST	TO WEST SANDIA DR (S)	25	8:03	1.1		
DRIVE ON	WEST SANDIA DR (S)	TO WEST SANDIA DR (E)	15	8:03	.1		
PICK UP ON	WEST SANDIA DR (S)	TO 32ND PL	5	8:05	.1	1	
PICK UP ON BOTH SIDES	32ND PL	TO	5	8:13	.1	7	3
DRIVE ON	32ND PL	TO WEST SANDIA DR (S)	15	8:13	.1		3
DRIVE ON	WEST SANDIA DR (S)	TO WEST SANDIA DR (E)	15	8:13	.1		3
PICK UP ON BOTH SIDES	WEST SANDIA DR (E)	TO 30TH ST	5	8:22	.1	7	6
PICK UP ON BOTH SIDES	WEST SANDIA DR (E)	TO 31ST PL	5	8:25	.1	3	8
DRIVE ON	31ST PL	TO	15	8:26	.1		8
PICK UP ON BOTH SIDES	31ST PL	TO WEST SANDIA DR (E)	5	8:35	.1	8	11
DRIVE ON	WEST SANDIA DR (N)	TO 33RD PL	15	8:35	.0		11
DRIVE ON	33RD PL	TO	15	8:35	.1		11
PICK UP ON BOTH SIDES	33RD PL	TO WEST SANDIA DR (N)	5	8:42	.1	6	14
DRIVE ON	WEST SANDIA DR (N)	TO 33RD PL	15	8:42	.0		14
DRIVE ON	33RD PL	TO	15	8:42	.1		14
PICK UP ON BOTH SIDES	33RD PL	TO WEST SANDIA DR (N)	5	8:52	.1	9	18
DRIVE ON	WEST SANDIA DR (N)	TO 35TH PL	15	8:52	.1		18
DRIVE ON	35TH PL	TO	15	8:52	.1		18
PICK UP ON BOTH SIDES	35TH PL	TO WEST SANDIA DR (N)	5	9:04	.1	11	23
DRIVE ON	WEST SANDIA DR (N)	TO 37TH PL	15	9:04	.0		23
DRIVE ON	37TH PL	TO	15	9:04	.0		23
PICK UP ON BOTH SIDES	37TH PL	TO WEST SANDIA DR (N)	5	9:11	.0	6	26
DRIVE ON	WEST SANDIA DR (N)	TO 39TH PL	15	9:11	.0		26
DRIVE ON	39TH PL	TO	15	9:11	.1		26
PICK UP ON BOTH SIDES	39TH PL	TO WEST SANDIA DR (N)	5	9:23	.1	11	31
DRIVE ON	WEST SANDIA DR (N)	TO 41ST PL	15	9:23	.1		31
DRIVE ON	41ST PL	TO	15	9:24	.1		31
PICK UP ON BOTH SIDES	41ST PL	TO WEST SANDIA DR (N)	5	9:33	.1	9	35
DRIVE ON	41ST PL	TO	15	9:33	.1		35
PICK UP ON BOTH SIDES	41ST PL	TO WEST SANDIA DR (N)	5	9:42	.1	8	39
PICK UP ON	WEST SANDIA DR (N)	TO 43RD PL	5	9:44	.1	1	39
DRIVE ON	43RD PL	TO	15	9:44	.1		39
PICK UP ON BOTH SIDES	43RD PL	TO WEST SANDIA DR (W)	5	9:56	.1	11	44
PICK UP ON BOTH SIDES	WEST SANDIA DR (W)	TO 45TH ST	5	9:58	.0	2	45
DRIVE ON	WEST SANDIA DR (W)	TO 47TH ST	15	9:59	.1		45
PICK UP ON BOTH SIDES	47TH ST	TO PENNSYLVANIA AVE	5	10:09	.1	9	49
DRIVE ON	PENNSYLVANIA AVE	TO 45TH ST	25	10:09	.1		49
DRIVE ON	45TH ST	TO 49TH LOOP	15	10:09	.0		49
PICK UP ON BOTH SIDES	45TH ST	TO WEST SANDIA DR (W)	5	10:17	.1	7	52
PICK UP ON BOTH SIDES	WEST SANDIA DR (W)	TO 47TH ST	5	10:21	.1	3	54
PICK UP ON BOTH SIDES	WEST SANDIA DR (W)	TO 46TH ST	5	10:23	.0	2	54
DRIVE ON	WEST SANDIA DR (S)	TO 44TH PL	15	10:24	.0		54
DRIVE ON	44TH PL	TO	15	10:24	.1		54
PICK UP ON BOTH SIDES	44TH PL	TO WEST SANDIA DR (S)	5	10:30	.1	6	57
BREAK TIME				10:30 TO 10:45			
DRIVE ON	44TH PL	TO	15	10:46	.1		57

ACTION			SPEED (MPH)	TIME (HR:MIN)	DISTANCE (MILES)	HOUSEHOLDS SERVICED	LOAD (PCT)
PICK UP ON BOTH SIDES	44TH PL	TO WEST SANDIA DR (S)	5	10:57	.1	10	62
DRIVE ON	WEST SANDIA DR (S)	TO 42ND PL	15	10:57	.1		62
DRIVE ON	42ND PL	TO	15	10:57	.1		62
PICK UP ON BOTH SIDES	42ND PL	TO WEST SANDIA DR (S)	5	11:08	.1	10	66
DRIVE ON	WEST SANDIA DR (S)	TO 40TH ST	15	11:08	.0		66
PICK UP ON BOTH SIDES	40TH ST	TO WEST SANDIA DR (N)	5	11:23	.1	14	73
DRIVE ON	WEST SANDIA DR (N)	TO 34TH ST	15	11:24	.1		73
PICK UP ON BOTH SIDES	34TH ST	TO WEST SANDIA DR (S)	5	11:38	.1	13	79
DRIVE ON	WEST SANDIA DR (S)	TO 38TH PL	15	11:38	.1		79
PICK UP ON BOTH SIDES	38TH PL	TO	5	11:45	.0	6	81
DRIVE ON	38TH PL	TO WEST SANDIA DR (S)	15	11:45	.0		81
DRIVE ON	WEST SANDIA DR (S)	TO 38TH PL	15	11:45	.0		81
PICK UP ON BOTH SIDES	38TH PL	TO	5	11:56	.1	10	86
DRIVE ON	38TH PL	TO WEST SANDIA DR (S)	15	11:56	.1		86
DRIVE ON	WEST SANDIA DR (S)	TO 46TH ST	15	11:57	.2		86
DRIVE ON	46TH ST	TO A ST	15	11:57	.1		86
DRIVE ON	A ST	TO	15	11:57	.0		86
PICK UP ON BOTH SIDES		TO	5	12:00	.0	2	87
DRIVE ON		TO A ST	15	12:00	.0		87
DRIVE ON	A ST	TO	15	12:00	.0		87
PICK UP ON BOTH SIDES		TO	5	12:04	.0	4	89
BREAK FOR LUNCH				12:04 TO 12:34			
DRIVE ON		TO A ST	15	12:34	.0		89
DRIVE ON	A ST	TO	15	12:34	.1		89
PICK UP ON BOTH SIDES		TO	5	12:39	.0	4	90
DRIVE ON		TO A ST	15	12:39	.0		90
DRIVE ON	A ST	TO	15	12:39	.1		90
PICK UP ON BOTH SIDES		TO	5	12:43	.0	4	92
DRIVE ON		TO A ST	15	12:43	.0		92
DRIVE ON	A ST	TO	15	12:43	.0		92
PICK UP ON BOTH SIDES		TO	5	12:48	.0	4	94
DRIVE ON		TO A ST	15	12:48	.0		94
DRIVE ON	A ST	TO PENNSYLVANIA AVE	15	12:49	.3		94
DRIVE ON	PENNSYLVANIA AVE	TO D ST	25	12:52	1.1		94
DRIVE ON	D ST	TO KIRTLAND ACCESS RD	30	12:53	.5		94
DRIVE ON	KIRTLAND ACCESS RD	TO WEST ORDINANCE RD	30	12:53	.3		94
DRIVE ON	WEST ORDINANCE RD	TO LAND FILL	30	12:55	1.0		94
UNLOAD				12:55 TO 13:10			
LEAVE LAND FILL				13:10			
DRIVE ON	WEST ORDINANCE RD	TO KIRTLAND ACCESS RD	30	13:12	1.0		
DRIVE ON	KIRTLAND ACCESS RD	TO RIDGECREST DR	30	13:14	.9		
PICK UP ON BOTH SIDES	RIDGECREST DR	TO PERIMETER DR SOUTH	5	13:16	.0	2	
PICK UP ON BOTH SIDES	RIDGECREST DR	TO FAIRCHILD AVE	5	13:20	.1	3	2
PICK UP ON BOTH SIDES	FAIRCHILD AVE	TO WALKER AVE	5	13:26	.1	5	4
PICK UP ON BOTH SIDES	WALKER AVE	TO GERRIS AVE	5	13:34	.1	8	8
PICK UP ON BOTH SIDES	WALKER AVE	TO HIRSCH DR EAST	5	13:40	.1	5	10
PICK UP ON BOTH SIDES	WALKER AVE	TO PERIMETER DR EAST	5	13:46	.0	5	12
PICK UP ON BOTH SIDES	PERIMETER DR EAST	TO SAN PABLO ST	5	14:19	.2	31	26
PICK UP ON BOTH SIDES	SAN PABLO ST	TO HIRSCH DR EAST	5	14:24	.1	4	28
PICK UP ON BOTH SIDES	SAN PABLO ST	TO GERRIS AVE	5	14:28	.1	4	30

ACTION		SPEED (MPH)	TIME (HR:MIN)	DISTANCE (MILES)	HOUSEHOLDS SERVICED	LOAD (PCT)
PICK UP ON BOTH SIDES SAN PABLO ST	TO FAIRCHILD AVE	5	14:33	.1	4	32
BREAK TIME			14:33 TO 14:48			
PICK UP ON BOTH SIDES SAN PABLO ST	TO RIDGECREST DR	5	14:53	.1	4	34
PICK UP ON BOTH SIDES RIDGECREST DR	TO FAIRCHILD AVE	5	15:26	.2	31	48
DRIVE ON FAIRCHILD AVE	TO WALKER AVE	15	15:27	.1		48
DRIVE ON WALKER AVE	TO GERRIS AVE	15	15:27	.1		48
PICK UP ON BOTH SIDES GERRIS AVE	TO SAN PABLO ST	5	15:58	.2	29	61
DRIVE ON SAN PABLO ST	TO HIRSCH DR EAST	15	15:59	.1		61
PICK UP ON BOTH SIDES HIRSCH DR EAST	TO WALKER AVE	5	16:30	.2	29	74
DRIVE ON WALKER AVE	TO PERIMETER DR EAST	15	16:30	.0		74
PICK UP ON BOTH SIDES PERIMETER DR EAST	TO PERIMETER DR SOUTH	5	16:36	.1	5	76
PICK UP ON BOTH SIDES PERIMETER DR SOUTH	TO RIDGECREST DR	5	17:13	.2	34	92
DRIVE ON RIDGECREST DR	TO KIRTLAND ACCESS RD	15	17:13	.0		92
DRIVE ON KIRTLAND ACCESS RD	TO WEST ORDINANCE RD	30	17:15	.9		92
DRIVE ON WEST ORDINANCE RD	TO LAND FILL	30	17:17	1.0		92
UNLOAD			17:17 TO 17:32			
DRIVE ON WEST ORDINANCE RD	TO KIRTLAND ACCESS RD	30	17:34	1.0		
DRIVE ON KIRTLAND ACCESS RD	TO D ST	30	17:34	.3		
DRIVE ON D ST	TO GARAGE	30	17:36	.9		

(The reverse of this page is blank.)

GLOSSARY

Air Force Refuse-Collection Scheduling Program: a set of four computer programs that schedule residential refuse-collection and produce printed schedules and maps of the routes.

map coordinate unit (MCU): the length, in inches, of a unit interval from the coordinate system appended by the user to the first map input to program RCINPT.

node: a numbered point on a street at which some characteristic of the street changes.

route: the travel and collection performed by one collection vehicle during a day, starting and ending at the garage.

section: a group of segments serviced by the same collection-vehicle trip.

segment: a portion of a street between two nodes.

(The reverse of this page is blank.)

INITIAL DISTRIBUTION

ADTC/CS	1
DDC/DDA	2
HQ AFSC/DL	2
HQ USAF/RDPS	1
AFIT/Library	1
AFIT/DE	1
AFIT/LSGM	1
National Science Foundation	1
EPA/ORD	1
USA-CERL/EH	1
USA Chief, R&D/EQ	1
USN Chief, R&D/EQ	1
AFETO/DEV	1
Hq AUL/LSE 71-249	1
Det 1 ADTC/TST	1
Det 1 ADTC/ECW	3
Det 1 ADTC/EC	1
USA-CERL/Library	1
USA-CERL	1
UNM-CERF	3

(The reverse of this page is blank)